2-26-2021 10:30 AM

A Physical Layer Framework for a Smart City Using Accumulative Bayesian Machine Learning

Razan E. AlFar, *The University of Western Ontario*

Supervisor: Bauer, Michael A, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Computer Science

Abstract

Smart cities are one of the most active research fields in the world, due to the various benefits and challenges associated with their implementation. A major challenge for smart cities is processing and transporting the huge volumes of data generated by the sensor network layer, which builds the fundamental physical layer. Ongoing research is needed to address the computational challenges arising in smart city environments, particularly to help ensure efficient operations around the sensor layer. To address this challenge, a novel physical layer framework is proposed that intelligently learns the behavior of the physical system to enable the agents to control the sensors effectively and achieve the predefined sensor(s) and environment objectives. In addition, a novel machine learning semi-supervised Bayesian learning algorithm is proposed to learn and predict the sensors' behaviors to efficiently manage the energy consumption in houses. The novel model and algorithm were used in various diverse simulations, where results demonstrated their effectiveness in managing the energy consumption in different house settings and smart city environments.

Keywords: Smart City, Sensor Network, Physical Layer, Internet of Things (IoT), Energy Consumption Management, Machine Learning (ML), Bayesian Learning, Semi-Supervised Learning.

Summary for Lay Audience

As cities continue to expand and the urban population grows, there is an increased need to provide new solutions to help humans live a better life and to boost productivity in cities. One solution includes integrating diverse smart devices and tools in cities to provide a smart city environment. However, the bigger the city, the more smart devices and tools are required for the smart city environment, and the more data is produced. The huge volume of data has to be processed and transported through a very important layer in smart cities called the physical layer, creating a challenge that can affect the efficiency of smart cities. To address this challenge, a novel physical layer framework is proposed that intelligently learns the behavior of the physical system in a smart city, which enables the smart devices around the city to be controlled effectively, achieving the pre-determined environment objectives.

Furthermore, controlling the energy consumption is vital for having a sustainable smart city, due to the lack of resources and the rapid growth of the world's population. A potential solution includes setting an energy consumption limit threshold for each house across the smart city, in order to try to manage the energy consumption. Another solution includes adding intelligence to the smart city, by utilizing a technique called Machine Learning (ML). ML is a collection of theories and methods that enable computers to learn from data and make predictions without being explicitly programmed. In this thesis, a novel ML-based algorithm is proposed that can learn and predict the sensors' behaviors to efficiently manage the energy consumption in houses, and keep the energy consumption below the limit threshold for each house while keeping citizens fulfilled. The novel model and algorithm were used in various diverse simulations, where results demonstrated their effectiveness in managing the energy consumption in various different house settings and smart city environments.

Acknowlegements

First and foremost, I would like to thank God the Almighty for granting me the patience, passion and resilience to accomplish this thesis successfully.

I would like to express my sincere gratitude towards my supervisor, Prof.Michael A. Bauer for providing me with this great research opportunity. I cannot thank him enough for his patience, mentorship, support, and guidance throughout the entire program.

I extend my gratitude towards the members of Western University and the Department of Computer Science for providing a great research environment, and making this degree possible.

I would also like to thank Dr.Yehia Kotb for his continuous help and support in revising my work, and for always being there as a great friend and mentor.

Furthermore, I would like to extend my heartiest gratitude to my parents: my father, Essam, who encouraged me to continue my studies and supported me through this whole journey and always believed in me. My mother, Ghada, my biggest supporter in life and the most loving soul I've ever known, I wouldn't be here today without your immense support, unconditional love and prayers. I'm forever thankful for everything you've done for me and still do.

To my siblings, Mohammad, Ismail, Ahmad and Rawan, you are the joy and blessings in my life. I couldn't embark on this journey without your continuous support and cheerful encouragements. I would also like to thank all my family members, friends and colleagues who were there for me and supported me.

Finally, special thanks goes to some of the great colleagues that I've met throughout this incredible journey. My experience wouldn't have been the same without your heartfelt support and valuable friendship.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

SMART cities are one of the most active research fields today. It aims to provide solutions to problems in urban environments, in order to help humans live better and to increase productivity in cities. Fifty-five percent (55%) of the world's population live in urban areas, and the proportion is projected to reach 68% by 2050, adding 2.5 billion people to urban areas [1]. Furthermore, the current smart cities market size in 2020 is valued at USD $98.95 billion and is projected to reach USD $463.89 billion by 2027, demonstrating a Compound Annual Growth Rate (CAGR) of 24.7% [2]. These values demonstrate a demand and need for effective smart cities.

Research into smart cities, like any new emerging topic, does not have a clear definition. One simple definition was proposed by Celino, *et al.* [3] which defined a smart city as a city that can effectively process networked information to improve the outcomes of any city operations aspects. It should be noted that this definition comes from the perspective of Internet computing only. The city operations mentioned by Celino, *et al.* [3] included, but are not limited to, providing information to authorities, e-commerce and businesses, energy optimization and control, resource production, resource consumption, traffic management, public safety, emer-

gency responses, and medical applications. Dameri, *et al.* [4] indicated that since the idea of a smart city is relatively a recent phenomenon, it suffers from ambiguity when it comes to a concrete definition. In addition, the spread of the smart cities trend worldwide across the different fields and industries make it very hard to come up with a universal smart city definition. The growth and expansion of such a new phenomenon is due to the vital problems related to urban life, which includes traffic, pollution, urban crowding, poverty and many more.

The word *smart* in smart city is not the only word used to describe the same concept. The phrase *intelligent city* is another phrase that has been introduced to refers to cities with several competencies. *Digital city* is another phrase used in the literature to emphasize the fact that a city is digitized, where it supports city wide data processing and information sharing. A third notion is that of the *sustainable city*. It basically describes an eco-friendly city, where reducing CO2 emissions with the help of technology is its main objective, along with efficient energy production and improving buildings' energy consumption.

A different term is *techno-city*, where technology is used to help infrastructure and services become more efficient and more functional. The focus in techno-city is on urban space quality, mobility, public transports, and logistics. One other term is the *well-being city* which is a city that aims to guarantee the best quality of life for citizens and be attractive for businesses [4]. For a comprehensive definition for the term *smart city*, all the elements discussed above should be taken into consideration. The following definition is introduced in this thesis:

Definition : A smart city is a well-defined city which utilizes advanced technologies, information, and resources to benefit the citizens globally, environmentally, economically, and socially. It exploits communications and sensor capabilities into the cities' infrastructures, optimizing electricity, transportation, and other supporting day-to-day life logistical operations, hence, enhancing the overall quality of life for everyone; it connects the physical infrastructure to leverage the intelligence of the city.

A large amount of the research around smart cities has also concentrated on deploying and using sensors [5], as well as on cloud and edge computing to leverage smart city applications, which includes Internet of Things (IoT) [6–8]. Furthermore, more research focused on handling big data and related analytics [9, 10], and on developing services and middleware to facilitate data handling [11, 12]. In addition, to add intelligence to smart cities, the use of Artificial Intelligence (AI) methods have been explored in smart cities applications [13]. The use of AI has also been driven by the need to effectively and efficiently manage smart city infrastructure consisting of IoT, large data, networking and the cloud and smart city applications [14]. Smart cities can be broadly represented by five significant components as illustrated in Figure 1.1.

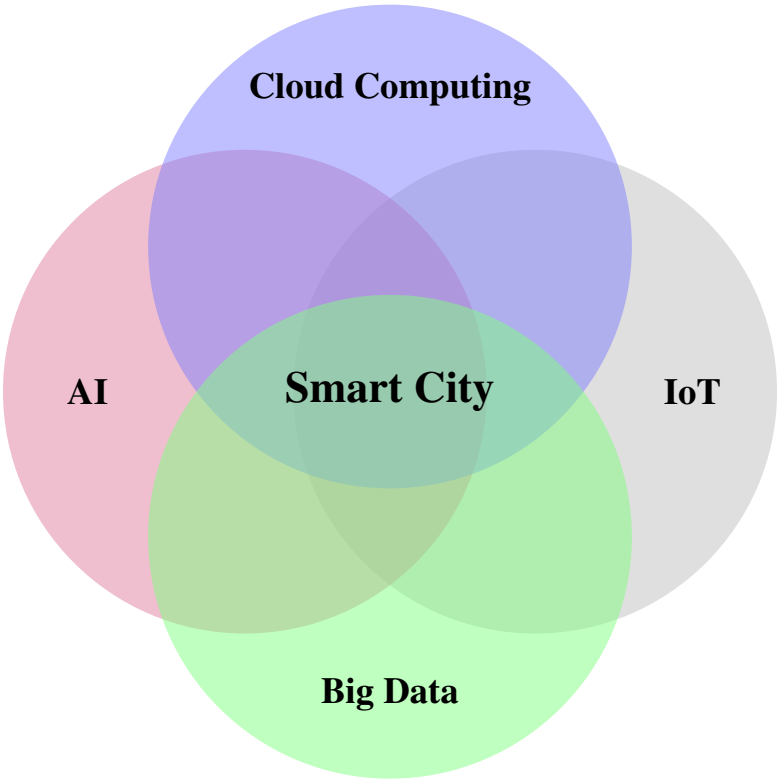


Figure 1.1: Venn Diagram of the Smart City Components

With the growth of IoT in smart cities, researchers began to focus on middleware to support sensors and related applications. The growing number of cities that are trying to exploit technologies and applications that provide expanded services and enhanced living standards for its citizens require software platforms to support development, deployment and use of smart city

infrastructures. For example, the Padova Smart City [15] initiative introduced a sensor network in the city of Padova, Italy. The platform collects environmental data, such as the temperature of the air and CO_2 emissions. The platform enabled the use of common protocols and data formats to allow interoperability among multiple city systems.

The *computational infrastructure* for a smart city can be broadly viewed in terms of four main layers (see Figure 1.2). Starting from the bottom of the hierarchy, the first computing layer is the physical layer, also referred to as the device layer, which is comprised of the sensors that generate the raw data and the actuators which can be used to change settings of devices, e.g. a thermostat. The layer above this is the IoT kernel layer, which collects data, monitors and controls the sensors and actuators. The third computing layer is the edge/fog layer, which enables computing, i.e., applications and services, to take place at the network edge. The last computing layer and at the top of the hierarchy is the cloud layer, which is the backbone of the IoT services and the initial receiver of all the information sent by smart devices. These four computing layers are shown in Figure 1.2. This infrastructure provides the foundation for the smart city applications and services. This can be monolithic software components, such as a database running in the cloud or fog computing system, or components of distributed applications and services or individual software components running on systems in the edge or IoT layer. One can think of the space of applications, services and their components as being overlayed on this computational infrastructure.

Smart city research faces many challenges, and those challenges vary from city to city and from one application to another. For example, applications that deal with citizen information face from privacy threats. These privacy threats cause many concerns where sectors, such as the financial or healthcare sectors, may not fully participate in smart city initiatives [3]. Other challenges are related to the nature of the data itself. How diverse is the data? How noisy is the data? What are the data cleaning processes for such a huge volume of data? The need to efficiently analyse collected data is one of the challenges faced by smart cities, particularly

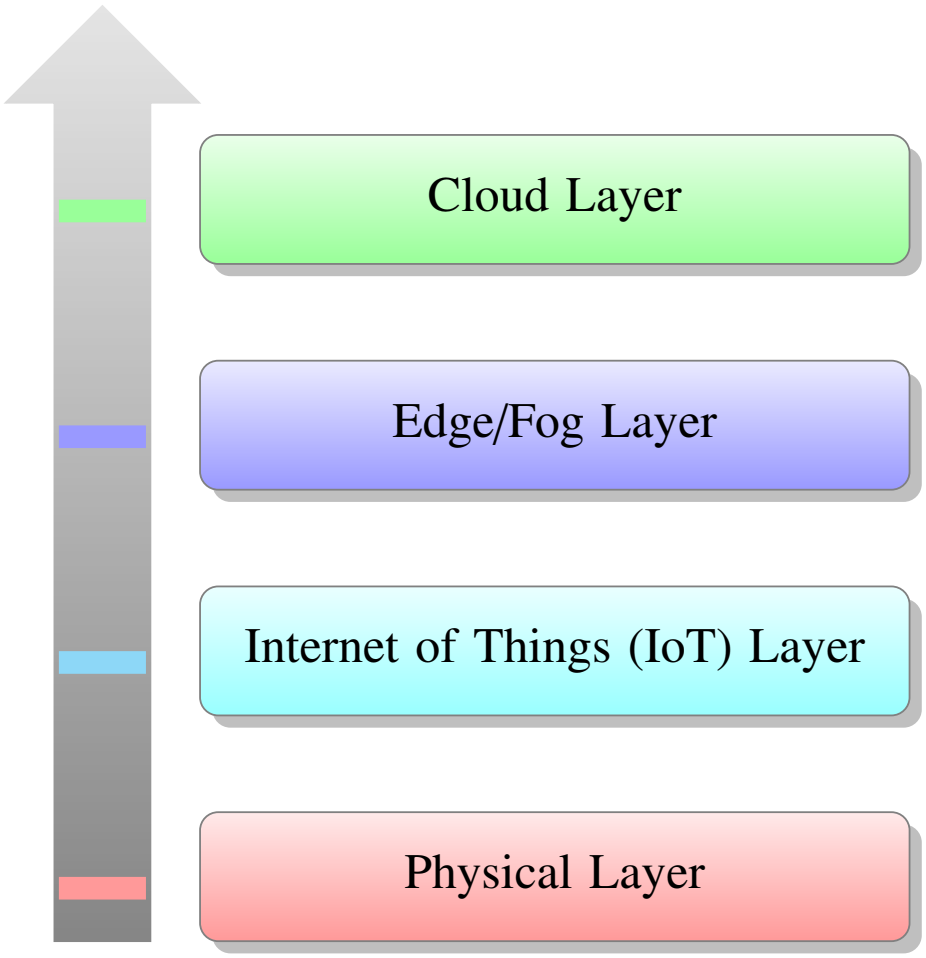


Figure 1.2: The Computing Layers of a Smart City

when there is a necessity for real-time responses and for an energy-efficient smart city.

1.1 Problem Statement

Smart cities are capturing the attention of researchers and citizens around the world. Their emergence ignited extensive research on the approaches to support the development of smart city applications, specifically around handling sensors and their data, as well as on the associated data analysis. With the complexity of smart city environments, namely the computational infrastructure and applications, ongoing research is needed to address the computational challenges arising in smart city environments, particularly to help ensure efficient operations.

The physical layer consists of devices such as sensors and actuators, where different devices serve different purposes and come in various shapes and sizes. The sensors can measure different aspects of the environment, such as temperature, humidity, air quality, or movement, whereas actuators can change their environment via movement and mechanism control. Smart cities heavily depend on sensors and actuators in order to get readings and enable applications to control houses, buildings, etc. The development of these types of smart city application requires methods and models that can facilitate the use of sensor data and enable use of actuators.

It is vital to be able to control the available resources to boost the efficiency in a smart city. In this thesis, these problems are addressed through a model of the physical environment and algorithms that can support the development of applications that utilize sensor data and leverage actuators to achieve operational objectives. The proposed physical framework can be utilized to address many different challenges in smart cities. One of those challenges is the energy consumption management. The utility of the model is illustrated and algorithms by using them to formulate a novel algorithm for addressing energy consumption.

In summary, the model of the physical layer takes into consideration the set of integrated sensors and the readings they produce. The model provides a framework for integrating the variables holding sensor readings, the role and use of intelligent agents, and the set of objectives to be met throughout during execution.

1.2 Thesis Contribution

A mathematical model is proposed for the physical layer, where the model is divided into two parts: the physical environment model, and the operational environment model. The model of the physical environment contains the sensors, variables in the environment that get values from sensor readings, agents that can take actions to adjust actuators and the objectives to

achieve in managing the physical environment and applications.

One of the challenges facing smart cities is energy consumption management, which this thesis tackles by proposing a novel algorithm based on the model of the physical environment. The algorithm utilizes machine learning techniques with probabilistic methods produce an even more powerful predictive mechanism [16]. The machine learning engine is used to analyze the historical data and to predict the future usage of smart homes. Bayes Network (BN), Naive Bayesian (NB), and Nearest Neighbor (NN) are examples of machine learning classification algorithms that are used for prediction [17]. In this thesis, a modified version of the Naive Bayes machine learning method is proposed. The original Bayesian classifier is a supervised learning-based algorithm, whereas the new proposed classifier is a semi-supervised learning algorithm. The algorithm starts as supervised learning, before turning into unsupervised learning when it can keep learning without the need for any supervision. The classification is done in two phases; phase one is supervised when the class label is provided along with input data, and phase two is unsupervised when no label is provided.

As noted, our novel physical layer framework that can be utilized to enable efficient operation of the sensors, etc., in a smart city. This includes algorithms for using data to choose actions to adjust sensors. Our target application is on energy consumption with the aim of optimizing and minimizing the energy consumption across a city. This is done by looking to limit the energy consumption of each house in a city. All the houses that exceed a certain limit of consumption are reduced gradually, in order to ensure that each house is not affected tremendously by the reduction and house occupants are satisfied. This is achieved by using the physical layer and machine layer approaches.

The learning algorithm learns the behavior of a sensor to predict the future actions that can be performed to reach a local or a global objective. The overall learning mechanism helps the agents learn how to reach a global objective. The main objectives are to optimize the infrastructure and services and to minimize the energy consumption in a city.

1.3 Thesis Overview

The remainder of this thesis is organized as follows. Chapter 2 presents an overview of smart cities, and some of the latest and most important work that has been done in managing the energy consumption in smart cities. Chapter 3 discusses preliminaries about the layers in a smart city, machine learning, and the proposed approach. Chapter 4 presents and describes the proposed mathematical model for the physical layer and the learning algorithms. Chapter 5 discusses the simulator used and demonstrates the results for the various experiments conducted. Finally, Chapter 6 concludes the thesis and discusses opportunities for future work.

Chapter 2

Literature Survey

In the early 1990s, the concept of smart city appeared for the first time. Technology, innovation, and globalization have been emphasized upon in the urbanization process by researchers [18]. Interest in smart cities has increased due to the growth of population in urban areas and due to the many challenges being faced in order to meet the citizens' needs in areas such as education, healthcare, security, water and many more. Smart cities aid in solving these issues by embracing the help of real-time sensors, cloud computing and big data analytics. In this thesis, a physical layer model is proposed which can tackle the energy consumption challenge faced in smart cities in order to boost the efficiency in a smart city. In this chapter, an overview of smart cities and related work in the field of energy consumption management in a smart city are discussed.

A smart city's infrastructure consists of sensors, data platforms, applications and security components. These interconnected components and their provided services are illustrated in Figure 2.1. However, the adoption of smart city solutions and services come with other challenges, such as ensuring the existence of sufficient communication infrastructure, available funds, required skills, and legal aspects to ensure that the privacy of citizens are not violated [19].

The sensor layer is an integral component in a smart city, as it incorporates both actuators and sensors. Actuators translate the received messages into actions, whereas sensors collect data by measuring the different aspects of the environment, such as temperature, humidity, air quality, water usage, or movement [19]. As a result, there are many different types of sensors that can be utilized for different functionalities. While applications derived from the collected data may make citizens' lives more convenient, the inclusion of additional sensors in the environment can increase the complexity of the physical layer, affecting the efficiency of smart cities. In addition, the inclusion of additional electrical devices in a smart city produce an energy consumption increase, creating challenges in tracking and maintaining the overall energy consumption in a smart city.

Figure 2.1: Smart City Components and Provided Services

Sensors have two types of network architectures: wired sensor networks and wireless sensor networks. The wired sensor network links the sensor directly to the device that will receive the input, by either using copper or fiber optic cables for the wiring [20, 21]. Wireless sensors are being widely used because of their efficiency, availability, increased mobility and low-

cost nature. Wireless sensor networks are often preferred to be used in various smart city applications, where they have been extensively used in applications such as collecting data to maintain property security and enhance quality of life [22]. The structure of wired and wireless sensor networks are shown in Figure 2.2.

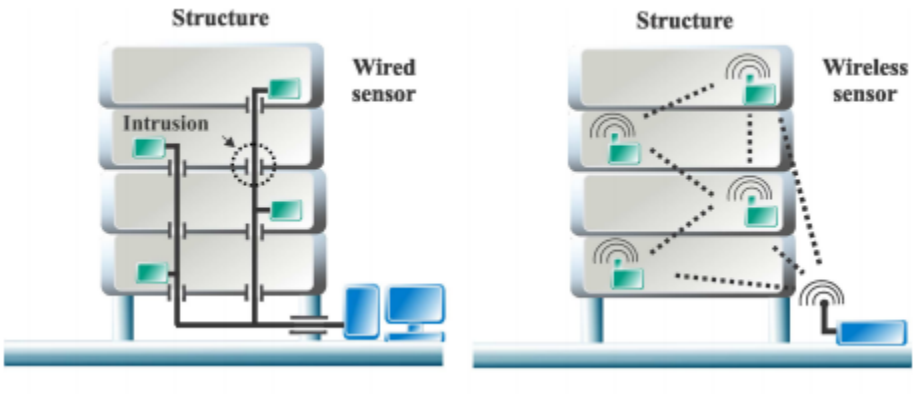


Figure 2.2: (a) Wired Sensor Network (b) Wireless Sensor Network

Sensor-based applications can provide real-time data to help city authorities with their operations [23]. The data collected from sensors can serve various objectives, such as health tracking, environmental monitoring, and prediction of disasters [24–27]. More complex sensors, such as accelerometers and cardio-tachometers have been employed by Wang, *et al.* [28] to detect accidental falls in elderly people. The authors in [29] presented a traffic light system that relies on a ZigBee wireless communication structure to provide a support infrastructure for intelligent control in smart cities. Furthermore, sensors have been used for automated applications, such as home automation [30] and industrial automation [31, 32].

Kang, *et al.* [33] analyzed various types of sensors and their importance from the perspective of smart home services. A sensor map tree was designed to help track the sensors and understand their behavior, as well as categorizing the smart home services. The results demonstrated that motion, temperature, video and humidity sensors were the most valuable sensors to use for smart home services.

Many platforms have been developed to manage sensors efficiently, as it is a vital element in

smart cities. Sentilo [34] is a platform that efficiently manages sensors and actuators, specifically implemented to support smart cities that require openness and interoperability. Sentilo controls the sensors and cloud computing using IoT notions, with the aim of sharing data with the applications. To ensure the scalability of the Sentilo platform, big data tools are utilized to gather and store data from sensors. Another platform was proposed by Piro, *et al.* [35], where a two-layered service platform was developed to originate smart city applications. The first layer manages the communication between the sensor network layer devices of the city, where the data is then collected from the devices using the second layer. Moreover, the second layer provides services for the development of applications that utilizes the smart city data.

Understanding, processing and deriving information from sensor data represents significant challenges. Therefore, optimization plays an important role in sensors. Sensors' optimization can be loosely classified into either a single-objective or a multi-objective optimization problem. The difference between the single-objective and the multi-objective optimization problem is the number of objectives to be targeted. When dealing with a single-objective optimization problem, the main goal is to maximize or minimize one objective at a time. Contrary to that, the main goal in a multi-objective optimization problem is to simultaneously optimize multiple objectives [36]. Objective optimization is an active research area due to its challenging and demanding task in smart cities [37].

In this thesis, the main target is to control the energy consumption in a smart city, specifically for buildings, as they are the largest consumers of energy in a smart city consuming an average total of 37% of the energy in developed countries [38]. Buildings include houses, structures, and offices. There has been several previous approaches in the literature that tackled the energy consumption challenge which will be discussed below.

Bhati, *et al.* [39] focused on the perception of Singapore households on its usage to save energy and reviewed various case studies. They concluded that in order a achieve an energy efficient smart home solution, AI modules should allow technology to seamlessly interact with con-

sumers, and smart meters should detect behavioral patterns and proactively take action, such as actively turning on/off a light if required.

Olatinwo, *et al.* [40] proposed a new approach to maximize the energy and throughput in sensors. They used wireless information and a power transfer method by harvesting energy from a dedicated radio frequency. The method was applied to water-quality monitoring, where the improved energy efficiency addressed the problem of energy scarcity. Nevertheless, the newly proposed algorithms were tested on a static environment only, not reflecting how smart cities operate dynamically in real-life.

Yoon, *et al.* [41] presented a system that can predict the efficiency of energy infrastructure within a smart city by using IoT and ML techniques. The proposed system used light sensors to collect information and predict efficient energy using Deep Neural Networks (DNNs). However, their results were limited, as a single sensor variable was used and the features of buildings and homes were not analyzed. Therefore, the system was not utilized in a variety of environments to validate its scalability.

Ejaz, *et al.* [42] focused on the energy management of appliances with high energy consumption in smart homes, by proposing a heuristic algorithm that scheduled smart home appliances while considering the tariffs and peak load. They utilized demand-side management that reduces the electricity cost by altering the system load. The approach worked well, however, the total load for both the optimum energy management and no energy management were the same at several times, demonstrating a major limitation.

In this thesis, a model for the physical layer is proposed, which combines various sensor readings to reduce the energy consumption in a smart city. The aim is to control the energy consumption in each house in a city, while ensuring that occupants are satisfied. This can be achieved by ensuring that the city adheres to a specific energy consumption limit. Moreover, if this energy consumption limit is exceeded, the individual houses that exceeded the limit will

be checked and their consumption will be reduced gradually. An NB-based ML approach is proposed to help in the decision making process and to try to achieve the global objective. Probabilistic forecasting has become more and more important to energy systems planning and operations [43]. The proposed approach combines multiple sensor readings in a large and dynamic environment, which none of the discussed previous approaches implemented, while attempting to keep citizens satisfied all the time.

Chapter 3

Preliminaries

In this chapter, the aim is to introduce and define the layers of a smart city, which comprise of the cloud layer, the fog/edge layer, the IoT layer, and the physical layer. Afterwards, the field of machine learning, alongside some of its relevant techniques and methods is introduced. Lastly, the proposed Bayesian Network Learning approach used in this work is described.

3.1 Smart City Layers

As mentioned earlier, a smart city is connected using different layers, where each layer performs a specific task and contributes immensely to the smart city's architecture. In the following subsections: the cloud layer, the fog/edge layer, the IoT layer, and the physical layer will be described thoroughly.

3.1.1 Cloud Layer

Cloud computing is defined as a "model for enabling convenient, on-demand access to a virtu-

alized shared pool of computing resources that can be rapidly provisioned and released with minimal management effort" [44]. The services delivered via cloud computing cover a vast range of options such as storage, servers, networking, applications, and processing power through to Artificial Intelligence (AI) and machine learning. Nowadays, any service that does not require an individual to be physically nearby the computer hardware that they are using can be delivered via the cloud. The location of the service is commonly irrelevant to the user, hence the term "cloud" is used to describe this layer.

The benefits of cloud technologies compared to other systems were highlighted by Shapiro, *et al.* [45] and Gordon, *et al.* [46] as follows:

- Cloud technologies are quick and can be deployed efficiently without the need for substantial IT experiences, or a large IT infrastructure.
- Cloud technologies are scalable as the IT infrastructure can be scaled up or down by the organizations depending on their need of consumption.
- Cloud technologies follow a multi-tenancy architecture, allowing the cost to be distributed effectively.

3.1.2 Fog/Edge Layer

Edge computing and fog computing terms are usually mixed up with each other due to their tasks similarities [47]. They both have the same main objective of bringing the intelligence and processing power closer to the data creation via the edge of the network, saving network bandwidth and reducing latency in services [48]. However, the main difference between them is the location where the intelligence and processing power are placed [49]. A fog environment places intelligence at the local area network (LAN), where the architecture transmits data from endpoints to a gateway, before transmitting it to sources for processing and return transmission.

On the other hand, edge computing places intelligence and processing power in IoT devices at the edge, rather than completely focusing on the infrastructure side. Figures 3.1 and 3.2 illustrate the architectures of fog and edge computing respectively.

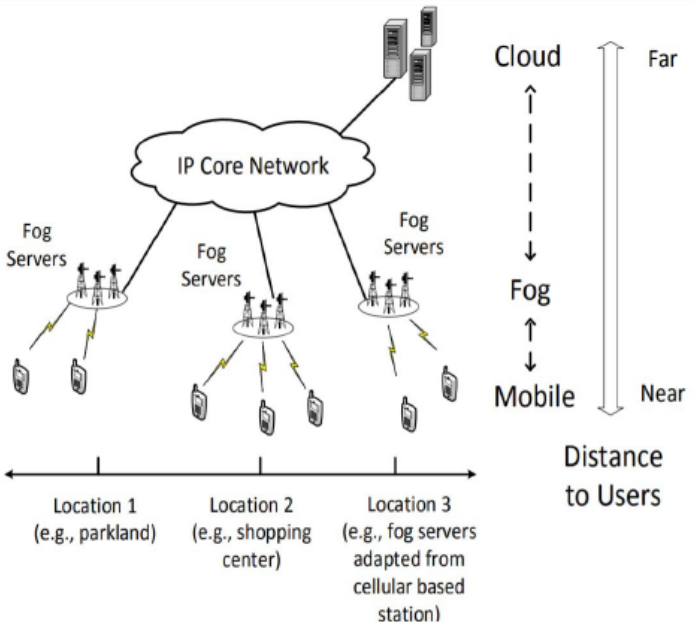


Figure 3.1: Architecture of Fog Computing

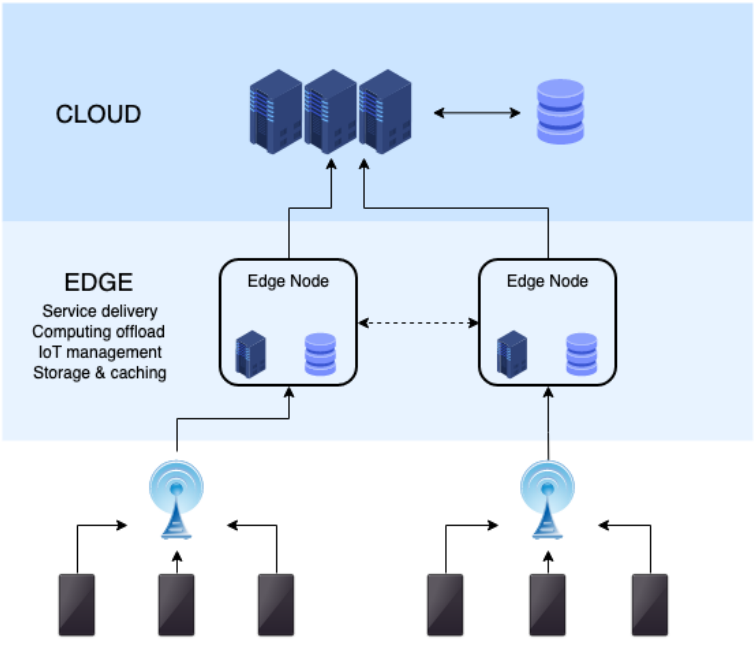


Figure 3.2: Architecture of Edge Computing

Fog Layer

The term ”Fog Computing” was introduced by Cisco in 2012 to extend the cloud computing paradigm to the edge of the network [50]. Verma, *et al.* defined the fog layer as a bridge between the IoT sensor layer and the cloud layer, where it is present at the gateway of the communication network, e.g., at UAVs and evacuation vehicles. The fog layer is capable to pre-process data in real-time because of its close proximity to the data origin and location-awareness [51]. As mentioned earlier, the fog layer utilizes a direct and short connection to serve mobile users, while the cloud connection uses a longer mobile connection. Hence, the fog nodes can be found closer to the user and on local service applications. As fog computing is deployed near the users, it can supply personalized and engaged location-aware services [52] which is seen as one of the most important characteristics of fog computing. Most of the proposed fog computing architectures are derived from the fundamental three-layer structure, which extends cloud computing, by introducing a fog layer between end devices and the cloud layer [53].

Edge Layer

IoT, AI, and stream data analytics edge-based applications are advancing rapidly, and their demand are expanding. Edge computing is becoming extremely significant for cloud computing as it allows the computation to take place at the network edge, on downstream data, and upstream data, representing the cloud services and IoT services respectively [54]. In edge computing, the computation and storage are performed on edge devices. To reduce the network latency and bandwidth between end devices and the cloud layer, computational tasks are performed by the edge devices. End devices act as data consumers and producers, therefore, these devices are able to request services from the cloud, and supply services back to the cloud [49].

3.1.3 IoT Layer

The term IoT (Internet of Things) refers to the rapidly growing number of digital devices in general, where the quantity of these devices is in billions now. The devices are utilized for communicating and interacting with others over the network/internet worldwide, where they are monitored and controlled remotely [55]. The IoT concept aims to make the Internet further immersive and widespread. Moreover, it enables easy access and interaction with a vast variety of devices such as home appliances, surveillance cameras, vehicles, monitoring sensors, actuators, displays, and so on. The IoT enhances the development of applications that utilize the huge amount of data generated by such objects providing new services to citizens, public administrations, and companies. [56].

The IoT uses the Internet to combine numerous heterogeneous objects. Accordingly, all objects that exist have to be connected to the Internet to provide ease of access. The motivation behind this is that smart cities contain sensor networks, and by connecting the intelligent appliances to the internet, their behaviour can be remotely monitored. Moreover, controlling appliances like refrigerators and washing machines by IoT makes houses offer better energy management [57]. Furthermore, to achieve this, sensors are utilized as they can be extended at various locations to collect and analyze data for utilization improvement [58].

3.1.4 Physical Layer

Physical layer consists of sensors and actuators. Sensors are extremely important in any smart application. They have the ability to detect any physical or chemical change, automating the application/devices to act smart after processing the collected data sensors. Sensors connect the physical world closely to the digital world in any IoT application, which can be implemented by supporting fog computing. By measuring and analyzing collected data to detect changes in physical objects, sensors play an essential role in the automation of any application. When-

ever any physical condition changes for which a sensor is created, it processes a measurable response. Figure 3.3 illustrates the sensing elements and the corresponding electrical signal.

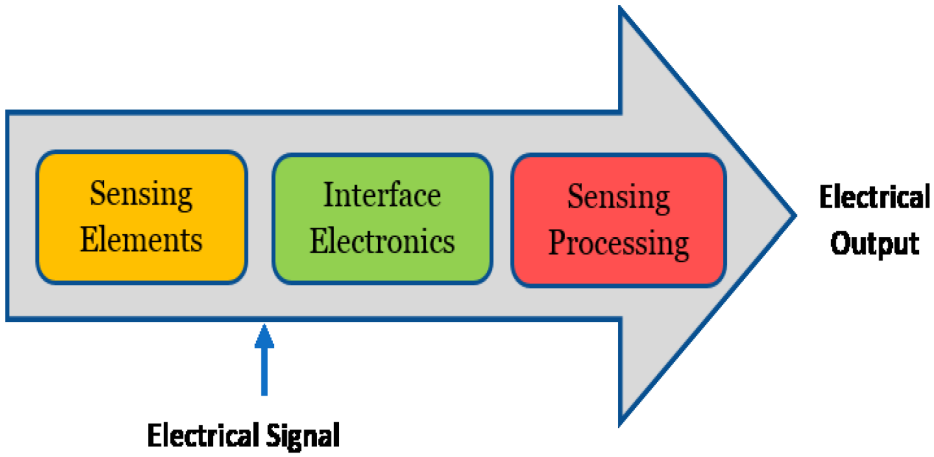


Figure 3.3: Sensing Elements and the Corresponding Electrical Signal

There are various kinds of sensors, where some are simple and some are complex. It is possible to identify sensors as dependent on their specifications, their method of conversion, the type of material used, the physical phenomena that it detects, the properties of what it tests, and the area of operation. Figure 3.4 shows different types of IoT sensors, where some of them will be explained more thoroughly below.

Temperature Sensors

Temperature sensors are useful in identifying the physical changes by estimating heat energy. Zafar, *et al.* [59] utilized temperature sensors to monitor the surrounding conditions of the environment, where the data collected was used to activate short term actions such as remotely controlling cooling or heating devices. A comparative sort of sensor was utilized by Nayyar, *et al.* [60] for smart agriculture, empowering farmers to expand their net yield and item quality by getting real-time information of their property.

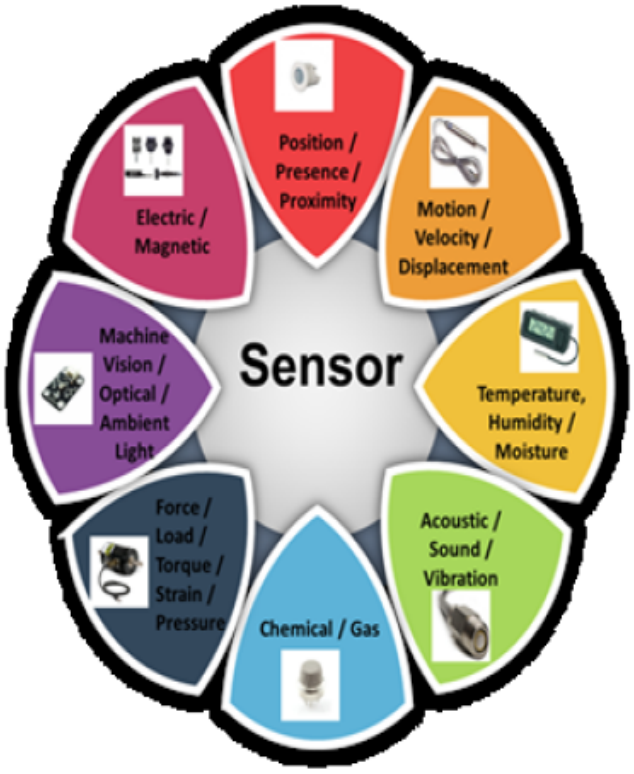


Figure 3.4: Different Types of IoT Sensors

Pressure Sensors

Pressure sensors convert the amount of force or pressure sensed into electrical signals. These type of sensors can be used in many applications, such as flow sensing, level sensing, and altitude sensing. As an example, Zhang, *et al.* [61] utilized pressure sensors in health monitoring to address devices interoperability issues.

Motion Sensors

Motion sensors are used to detect physical and kinetic movements in the surrounding. These type of sensors can be used in many applications and services and are commonly utilized in homes to provide monitoring and surveillance. Ansari, *et al.* [62] used motion detection sensors to provide a security alarm system using low processing chips, where photos and videos were

sent to a cloud server when motion was detected.

Humidity Sensors

Humidity sensors are used to measure the air temperature and moisture to signal humidity in the environment. These type of sensors are commonly used in the medical, automobile, agriculture, and manufacturing industries. As an example, Deekshath, *et al.* [63] proposed a real-time application that utilized humidity sensors for smart agriculture, increasing the farmers' overall yield and improving the output product quality.

Infrared Sensors

Infrared sensors are used to measure heat emissions, as well as transmitting or identifying infrared radiations to sense certain qualities of specific objects. These type of sensors are commonly utilized for home automation, smart security [64], smart parking [65], and waste collection systems [66].

3.2 Machine Learning

ML is a branch of AI focused on building applications that learn from data and improve their accuracy over time without being explicitly programmed to do so. An ML algorithm is trained to find structures and features in enormous amounts of data, rather than just a sequence of statistical processing steps, in order to make decisions and predictions based on new data.

ML is extensively used in a variety of applications and fields today, such as in self-driving cars, effective recommender systems, intelligent digital assistants, understanding the human genome, and much more.

There are four basic steps for building an ML application or model. First, a training data set is selected and prepared, followed by determining the algorithm to run on this prepared training data set. Afterwards, the algorithm is trained to create the model. Finally, the created model is used with new data and continuously improved upon for enhanced accuracy and effectiveness over time.

There are three main machine learning methods: supervised learning, unsupervised learning, and semi-supervised learning [67].

3.2.1 Supervised Learning

Supervised learning is where an algorithm learns a mapping function from the input variable(s) to the output variable(s), where the goal is to approximate the mapping function so well for a high accuracy prediction of the output when you have new input data. As an example, Ullah, *et al.* [68] utilized a supervised machine learning approach for lightning detection in a smart system, where the algorithm was given both the inputs and outputs.

Supervised learning usually requires less training data than other machine learning methods as the results of the model can be compared to the actual labeled results. However, it is expensive and difficult to prepare correctly labeled data and there is the danger of creating a model that is biased to the training data and does not generalize to new data.

3.2.2 Unsupervised Learning

Unsupervised learning is where an algorithm extracts meaningful and interesting features from the inputs when you only have input variables and no corresponding output variables, where the goal is to model the underlying structure and patterns in the data to further understand and learn about the data.

As an example, Kim, *et al.* [69] proposed an unsupervised machine learning algorithm to detect anomalies in time series data for edge computing in industrial IoT, where the algorithm was only given the inputs. Unsupervised learning often possess less computational complexity than supervised learning, as one is not required to label and understand the data inputs and it is usually easier to get unlabeled data. However, results could be less accurate and cannot be ascertained due to the lack of prior knowledge.

3.2.3 Semi-supervised Learning

Semi-supervised learning sits right in between both supervised and unsupervised learning, where you have a large amount of input data and only some labeled output data. The goal is to use a smaller labeled data set to guide classification, extract features, and learn structures from a larger, unlabeled data set, solving the difficulty and time-consumption challenge of correctly labeling data that supervised learning may face.

As an example, Oh, *et al.* [70] utilized surface mounted audio sensors in combination with machine learning algorithms to detect faults in a complex sound machine using a semi-supervised approach. Many real world machine learning problems fall into this area, as collecting unlabeled data is relatively cheap and easy, and a mixture of supervised and unsupervised techniques can be used.

3.3 Proposed Approach

The objective in this thesis is to manage the sensor network layer to bring down the overall computational load in a smart city. The goal is to understand the sensors behavior and to understand the surrounding environment, as well as to be able to efficiently enhance the performance of the smart city layers. To achieve this goal, a probabilistic, semi-supervised Bayesian

learning approach was utilized.

This approach handles uncertainties effectively and improves the decision making process, as well as reduces the need and cost of collecting labeled data. A preliminary definition of the Bayesian learning approach is given in the next subsection, and a more detailed discussion is found in Chapter 4.

3.3.1 Bayesian Network Learning

Bayesian networks were introduced by Judea Pearl in 1980s with the development of AI, enabling probabilistic beliefs to be systematically and locally assembled into a single, coherent whole [71]. Bayesian provide a structured, graphical representation of probabilistic relationships between several random variables and an explicit representation of conditional independencies via directed acyclic graphs (DAGs). For example, a Bayesian network can represent the probabilistic relationships between various different features and targets, where the probabilities of the presence of these targets are computed given the features. In this work, Bayesian networks will learn to model the behavior of the sensors to be able to control the sensor's energy consumption. Due to their mathematically grounded framework, Bayesian networks are the most popular method for uncertain expert knowledge and ratiocination, where it is vastly applied in large number of research areas [72].

Bayesian networks make use of Bayes Theorem during inference and prior to learning. Bayes Theorem is an approach for calculating the conditional probability of an event, and is defined as:

$$Posterior = \frac{Likelihood * Prior}{Evidence} \tag{3.1}$$

To understand these terms better, let us discuss them in terms of the probabilities of two events, namely A and B. Posterior refers to the probability of event A occurring given event B, and Prior refers to the probability of event A occurring. Furthermore, Likelihood refers to the probability of B occurring given event A, and Evidence refers to the probability of event B occurring.

Chapter 4

Physical Layer Model

In this chapter, a mathematical model for the physical layer $\aleph$ is presented. The physical layer model $\aleph$ is divided into two sections: the physical environment $\aleph_p$, and the operational environment $\aleph_o$. The physical environment $\aleph_p$ consists of the environment $\mathbb{E}$, the sensors $\mathbb{S}$, and the actions Λ, whereas the operational environment $\aleph_o$ consists of the agents $\mathbb{A}$, the reading history $\mathbb{H}$, and the objectives $\mathbb{O}$. The learning algorithms using NB classifiers will be introduced and explained as well. Moreover, a new modified Bayesian for accumulative learning algorithm will be explained, which is an enhanced version of the NB classifier.

As mentioned earlier, the physical layer $\aleph$ is built out of the physical model $\aleph_p$, and the operational model $\aleph_o$. Mathematically, it is described as the following tuple:

$$\aleph = \langle \aleph_p, \aleph_o \rangle \tag{4.1}$$

4.1 Physical model $\aleph_p$

Physical layer model $\aleph_p$ is the mathematical model for the physical layer. Mathematically, $\aleph_p$ is described as follows:

$$\aleph_p = \langle \mathbb{E}, \mathbb{S}, \Lambda \rangle \tag{4.2}$$

where,

1. $\mathbb{E}$ is the environment.
2. $\mathbb{S}$ is the set of sensors in $\aleph_p$.
3. Λ is the set of actions that are applied to the environment.

The following sections explain each component that belongs to $\aleph_p$ in details:

An environment $\mathbb{E}$ is defined to be a set of variables, where each variable takes on values that describe a certain natural phenomena, such as temperature, pressure or humidity. Sensors $\mathbb{S}$ provide those reading values to the set of variables over time. The number of variables depends on the number of sensors that exist in $\aleph_p$. Each variable represents a type of a measurement for a physical phenomenon. Mathematically, the environment is modelled as follows:

$$\mathbb{E} = \{V_1, V_2, \ldots, V_n\} \tag{4.3}$$

where,

1. $n = ||\mathbb{E}||$
2. $V = \{v_1, \ldots, v_k, v_i \in V\}$ is the set of variables associated with the sensor, and each V_i is associated with one $s_j \in \mathbb{S}$.

The set of sensors $\mathbb{S}$ is a vital component in $\aleph_p$ since it provides readings to describe the environment over time. This is done by providing environment variables with values as mentioned previously. A sensor is modelled as follows:

$$s_i \in \mathbb{S} = \langle V, \Psi_i, \tau, \Phi(t) \rangle \tag{4.4}$$

where,

1. V is the set of variables associated with the sensor as defined previously.

2. $\Psi_i = \{\Psi_{i1}, \ldots, \Psi_{ik}\}, \Psi_{ij}(v_j \in V)$ is a transfer function that maps a variable v_j of the sensor to a value r_t at time t; r_t is a numeric value that belongs to $\Re^+$; Ψ_{ij} describes sensor behavior; there is a transfer function for each variable.

3. τ is the rate of readings. In other words, $\tau_i = \frac{1}{f_i}$, where f_i is the frequency of the sensor s_i.

4. $\Phi(t) = \langle r_1, r_2, \ldots, r_i \rangle$, is the sensor state which is the tuple of values assigned to $v \in V$ at time t and $r_i = \Psi(v_i)$, where i is the number of states for each variable.

As described, it is assumed that at any point of time, a sensor s_i can provide a reading for each variable associated with that sensor. Consequently, at time t, an environment state $\mathbb{E}_t$ can be defined as:

$$\mathbb{E}(t) = \langle r_1, r_2, \ldots r_n \rangle \tag{4.5}$$

where $r_i = \Psi_i(v_i)$, and Ψ_i is the transfer function defined previously. We can think of $\mathbb{E}(t)$ as an *instance* of the environment $\mathbb{E}$ at some point of time t. It is worth noting that if there is a sensor that deals with four different variables, then it is going to have four transfer functions inside it.

As mentioned above, for every sensor $s_i \in \mathbb{S}$, there exists a sensor state Φ at time t. A sensor state $\Phi(t)$ is the set of readings assigned to sensor variables at time t. Note that:

$$\Phi(t) \subseteq \mathbb{E}(t) \tag{4.6}$$

At any point of time, environment state $\mathbb{E}(t)$ is the joining of all sensor states at time t.

An action is any activity applied to $\aleph_p$ to achieve an objective. As will be seen later in this chapter, objectives are classified as either global objectives which represent an overarching strategy for the physical model or local objectives for a sensor. It should be emphasized that objectives may change based on the changes that occur in the environment states. For example, if the temperature decreases, then the objective will change accordingly.

In this thesis, it is assumed that a finite set of basic actions is predefined; a basic action is an action that cannot be divided any further. An action changes the environment from one state $\mathbb{E}(t_1)$ to another state $\mathbb{E}(t_2)$. An action is applied to a sensor in order to change the value of a variable in order to change the state of the environment. The set of actions is finite. Actions are represented as follows:

$$\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_l\} \tag{4.7}$$

where $l = \|\Lambda\|$.

Examples of actions would be:

1. Changing the desired humidifier setting to increase or decrease humidity levels gradually.

2. Changing the desired thermostat setting to increase or decrease temperature levels gradually.

4.2 Operational model $\aleph_o$

The operational model $\aleph_o$ is the representation of how decision making processes for achieving objectives for the physical environment are modeled, as well as how the reading history $\mathbb{H}$ is stored. The operational model $\aleph_o$ is mathematically modelled as follows:

$$\aleph_o = \langle \mathbb{H}, \mathbb{O}, \mathbb{A} \rangle \tag{4.8}$$

where,

1. $\mathbb{H}$ is the reading history of sensors.
2. $\mathbb{O}$ is the set of objectives that $\aleph_o$ needs to reach.
3. $\mathbb{A}$ is the set of intelligent agents corresponding to each sensor.

The description of these elements will be illustrated with more details in the subsections below:

4.2.1 Reading History $\mathbb{H}$

History $\mathbb{H}$ is the previous readings that occurred in the past. Theoretically speaking, $\|\mathbb{H}\|$ could be infinite, however, in reality and for the sake of performance, and due to the limited memory capacity of embedded environments, history length has to be controlled.

For every sensor, there is a history. Since every sensor has one or more variables, history keeps track of the sequence of readings per variable. History is mathematically represented as:

$$\mathbb{H} = (\langle t_0, v_1, r(t_0) \rangle, \ldots, \langle t_0, v_k, r(t_0) \rangle, \ldots, \langle t_\tau, v_1, r(t_\tau) \rangle \ldots \langle t_\tau, v_k, r(t_\tau) \rangle) \tag{4.9}$$

Equation 4.9 shows how the history is stored. It shows that the history is a sequence of tuples. Every tuple consists of the timestamp, the variable and the reading of that variable. Note that at any time, the timestamp can have a single tuple for a single variable, tuples for some variables, tuples for all variables, or no tuples at all. It is a sequence of tuples sorted according to time t.

4.2.2 Objectives $\mathbb{O}$

The set of objectives $\mathbb{O}$ is being passed to the operational model $\aleph_o$ in order to be met. An operational model $\aleph_o$ might have one or more objectives to achieve.

In this thesis, objectives are classified into two types:

1. Global Objectives $\mathbb{O}_G$
2. Local Objectives $\mathbb{O}_L$

Mathematically, it can be described as:

$$\mathbb{O} = < \mathbb{O}_L, \mathbb{O}_G > \tag{4.10}$$

Objectives are complex Boolean expressions that the operational system tries to make true. The formation of the objective depends on whether it is a global or a local objective. The formation of a global objective would be something like $e \leq e_m$, where e is the total energy consumption and e_m is the maximum amount of energy consumption that the city cannot exceed. e is computed by calculating the total energy consumption in a house in kilowatt-hours (kWhs), taking into consideration the number of rooms, and the number of occupants in each house. Global objectives are set on the cloud layer.

The local objectives are set on the edge layer. If there is a sensor with three variables, v_t for

temperature, v_h for humidity, and v_il for illumination, then one objective of the sensor could be something similar to:

$$t_l \leq v_t \leq t_h \text{ and } h_l \leq v_h \leq h_h \text{ and } il_l \leq v_{il} \leq il_h$$

.

where,

1. v_t is the variable having temperature readings.
2. t_l is the minimum temperature in the objective.
3. t_h is the maximum temperature in the objective.
4. v_h is the variable having humidity readings.
5. h_l is the minimum humidity in the objective.
6. h_h is the maximum humidity in the objective.
7. v_{il} is the variable having illumination readings.
8. il_l is the minimum illumination in the objective.
9. il_h is the maximum illumination in the objective.

4.2.3 Intelligent Agent $\mathbb{A}$

Modern and sophisticated applications introduce the need to take IoT a step further to become what is known to be the Internet of Intelligent Things (IoIT) [73]. IoIT adds intelligence to "things" in a smart city. This reduces the need for intensive communication with the fog layer or with the cloud layer, since much of the learning and decision making is handled in the IoT

layer. In this thesis, for this to be achieved, it is assumed that for every sensor $s \in \mathbb{S}$ there is a corresponding agent $\alpha \in \mathbb{A}$, where s and α are associated; it is also assumed that a single agent handles all the variables that a sensor provides readings for. Agent α tries to meet an objective $o \in \mathbb{O}_L$ by changing an environment state $\mathbb{E}(t)$ at time t.

Agents are executable software that learn and take decisions. They run on the IoT or edge layer on devices such as cellphones, computers, Raspberry Pis, and so on. They do not run on sensors but they read from them and control the actuators. The sensors will be connected to the edge devices on the edge layer where the agent runs. Agents read from the environment using sensors and act on the environment using the actuators.

An intelligent agent $\alpha \in \mathbb{A}$ is mathematically described as follows:

$$(\alpha \in \mathbb{A}) = \langle \Xi, \Omega, \Upsilon, \Lambda_\alpha \subseteq \Lambda, s_\alpha \in \mathbb{S} \rangle \tag{4.11}$$

where,

1. Ξ is the learning algorithm that agent α uses to learn about the behaviour of the associated sensor $s \in \mathbb{S}$.

2. Ω is the prediction algorithm that predicts the future readings of each v in a sensor s.

3. Υ is the action selection process that selects the suitable actions for each v in a sensor s. The selection is based on the prediction $\wp \in \Omega$ and the objective $o \in \mathbb{O}_L$.

4. $\Lambda_\alpha \subseteq \Lambda$ is the set of actions that can be taken by agent $\alpha \in \mathbb{A}$.

5. $s_\alpha \in \mathbb{S}$ is the sensor associated with the agent.

Every agent can have one or more objectives depending on the number of sensor variables the agent needs to change. The granularity of agents with respect to sensors in this thesis is one to

one. In other words, every agent is responsible for a sensor that might have one or more sensor variables to avoid having a single point of failure. A single point failure is the case when a centralized system is controlled by a single agent, where the whole system fails if the agent fails. It is worth noting that the global objective $\mathbb{O}_G$ is announced to all agents in order for them to start working locally and autonomously towards the achievement of that objective.

Figure 4.1 represents a data flow explaining how the mathematical model relate to the smart city computational layers discussed previously.

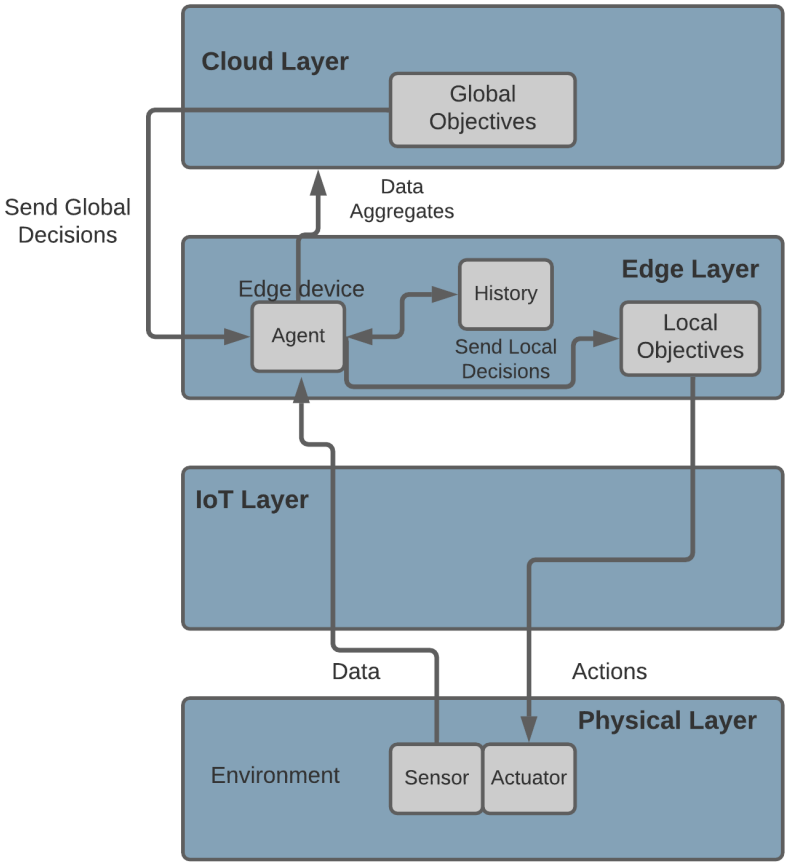


Figure 4.1: Data Flow

The following subsections describe the learning algorithm Ξ, the prediction process Ω, and the action selection process Υ. The learning algorithm Ξ is achieved through an NB classifier. The learning process goes through a few steps as will be seen in the rest of the section. Next,

the proposed modified Bayesian for accumulative learning is discussed, which is an enhanced version of the NB classifier presented.

The Process of Learning

In this subsection, the motivation and discussion of the learning process is explained in detail. First, the problem is probabilistic since it depends on predicting and estimating future actions that can be taken. More specifically, the NB algorithm is considered since it is computationally efficient and is able to achieve good results on a wide range of problems. The learning algorithm is general and can be used in many applications, however, it will be used in controlling the energy consumption in a city in this thesis.

The prediction helps us estimate future values of variables. This is important since we do not want to exceed a certain limit of any given objective. If we are able to predict future values, then situations can be handled better and objectives and limitations will not be exceeded. In this thesis, a new modified Bayesian for accumulative learning algorithm is presented, which is an improved version of the NB classifier. The algorithm is explained in Section 4.3.

a) Statistical Summary of the Data

The data used in our NB algorithm consists of sensor variables and their values. The data is initially separated into classes and the mean and the standard deviation of the data is calculated to be able to get the probability. Algorithm 1 describes how the data set is summarized. It takes in a set of data and returns the mean μ and standard deviation σ per column. For descriptive purposes, the data is organized into columns as an input to the algorithm.

The algorithm works as follows:

Algorithm 1 : $\mathbb{D}$ Summary

Input : data set $\mathbb{D}$.
Output : $\mathbb{S}tats$ data statistics.

Begin
$rCount \leftarrow 0$
for all $d \in \mathbb{D}$ **do**
$cIndex \leftarrow 0$
for all $c \in d$ **do**
$\mu[cIndex] \leftarrow \mu[cIndex] + c$
$cIndex \leftarrow cIndex + 1$
end for
$rCount \leftarrow rCount + 1$
end for
$c \leftarrow 0$
while $c < cIndex$ **do**
$\mu[c] \leftarrow \frac{\mu[c]}{rCount}$
$c \leftarrow c + 1$
end while
for all $d \in \mathbb{D}$ **do**
$cIndex \leftarrow 0$
for all c **do**
$\sigma[cIndex] \leftarrow \sigma^2[cIndex] + (d[cIndex] - \mu[cIndex])^2$
$cIndex \leftarrow cIndex + 1$
end for
$\sigma[cIndex] \leftarrow \frac{\sqrt{\sigma[cIndex]}}{rCount}$
end for
return (μ,σ,$rCount$,$cIndex$)
End.

1. First, it iterates over each element of each column in the data set and gather all of the values for each column into a list.

2. The mean and the count of the list are then calculated.

3. Afterwards, the variance is calculated, and the standard deviation is outputted by taking the square root of the variance.

4. The statistics are gathered into a list of tuples of statistics, and the operation is repeated for each column in the data set.

5. At the end, the summary of the data set is generated, where the mean, standard deviation, and count of each column are calculated, and a list of tuples of statistics is returned.

b) Classification of Data $\mathbb{D}$

The classification of the data set is an essential component of the learning process. In Algorithm 1 the summary statistics for each column is calculated, where the statistics will be used after the classification.

Algorithm 2 is the classification of the data set $\mathbb{D}$, and it processes as follows:

1. First, it iterates over the data set and split it by class.

2. Every class is assigned a specific label, and every data item is checked to which class it belongs to.

3. Next, if a data item belongs to a class that has not been labelled yet, then a specific label is created for that class, and the corresponding data item is added to that class.

4. Lastly, the statistics are calculated for each class.

Algorithm 2 : $\mathbb{D}$ Classification:

Input : data set $\mathbb{D}$.
Output : $\mathbb{C}lasses$ and $\mathbb{S}$ummary .

Begin
$\mathbb{C} \leftarrow \emptyset$
$cCount \leftarrow 0$
for all $d \in \mathbb{D}$ **do**
4: $label \leftarrow \mathbb{C}.createLabel(d)$
$c \leftarrow \emptyset$
if $label \notin \mathbb{C}$ **then**
$c \leftarrow newClass()$
8: $c.label \leftarrow label$
else
$c \leftarrow \mathbb{C}.getClass(label)$
end if
12: $\mathbb{C} \leftarrow \mathbb{C} \cup d$
$\mathbb{C} \leftarrow \mathbb{C} \cup c$
end for
$\mathbb{S}tats \leftarrow \emptyset$
16: **for all** $c \in \mathbb{C}$ **do**
$\mathbb{S}tats \leftarrow \mathbb{S}tats \cup Summary(c)$
end for
return $(\mathbb{C}, \mathbb{S}tats)$
End.

c) The Prediction Process

The statistics calculated from our training data can now be used to calculate the probabilities of the new data. Calculating the probability or likelihood of a real value can be challenging. Hence, it can be achieved by assuming that the values are drawn from a Gaussian distribution. This Gaussian distribution will have the mean and standard deviation values that were calculated in Algorithm 2.

Next, class predictions are performed on new data items by calculating the probability of each new data item belonging to each particular class, producing a list of probabilities. After an agent $\alpha \in \mathbb{A}$ learns from history $\mathbb{H}$, agent $\alpha \in \mathbb{A}$ is ready for predicting the future behavior of readings. Algorithm 3 is the process of class prediction using probabilities calculation, and it processes as follows:

Algorithm 3 : Class Probabilities

Input : $\mathbb{S}$ummaries and d.
Output : $\mathbb{P}$redicted $\mathbb{P}$robability.
Begin
$ClassProbability \leftarrow \emptyset$
$index \leftarrow 0$
for all $summary \in \mathbb{S}ummaries$ **do**
$\mu \leftarrow summary.\mu$
$\sigma \leftarrow summary.\sigma$
$\mathbb{P} \leftarrow 1$
$cindex \leftarrow 0$
for all $c \in d.Columns$ **do**
$exponent \leftarrow e^{\frac{-(c-\mu)^2}{2\times\sigma}}$
$v \leftarrow \frac{1}{\sqrt{2\times\pi\times\sqrt{\sigma}}} \times exponent$
$\mathbb{P} \leftarrow \mathbb{P} \times v$
end for
$ClassProbability[index] \leftarrow \mathbb{P}$
$index \leftarrow index + 1$
end for
return ($ClassProbability$)
End.

1. First, it iterates over the class summaries, which include the mean and standard deviation of each class.

2. Afterwards, it iterates over each column in the data set.

3. The probability of each column in every class is calculated by using the Gaussian probability distribution function.

4. The set of class probabilities are calculated and returned for every row belonging to each particular class.

Afterwards, the class that gives the highest probability will be assigned as the prediction to that specific data item. Algorithm 4 represents the prediction process, and it operates as follows:

Algorithm 4 : Prediction Process:

Input : Summaries and Row.
Output : Best Label.
Begin
$\mathbb{P} \leftarrow ClassProbabilities(Summaries, Row)$
Best$\mathbb{L} \leftarrow \emptyset$
Best$\mathbb{P} \leftarrow -1$
$index \leftarrow 0$
$chosenindex \leftarrow -1$
for all $p \in \mathbb{P}$ **do**
  if $p >$ Best$\mathbb{P}$ **then**
    $chosenindex \leftarrow index$
    Best$\mathbb{P} \leftarrow p$
  end if
  $index \leftarrow index + 1$
end for
Best$\mathbb{L} \leftarrow summaries[chosenindex].label$
return (Best$\mathbb{L}$)
End.

1. This algorithm iterates over the class probabilities that were calculated in Algorithm 3, and the class that produces the highest probability for the particular row is selected.

2. At the same time, the label for that class is saved.

3. At the end, the best class probability and its label for the desired item is returned.

Lastly, the NB algorithm is introduced. NB is a classification algorithm for binary (two-class) and multi-class classification problems. It is called *Naive* because the calculations of the probabilities for each class are simplified by eliminating the denominator of Bayes Theorem to make their calculations tractable. In addition, they are assumed to be conditionally independent given the class value. Algorithm 5 represents the NB algorithm, and it operates as follows:

Algorithm 5 : Naive Bayes Algorithm:

Input : Data set $\mathbb{D}$.
Output : Predictions.
Begin
$\mathbb{S} \leftarrow Classification(D.train)$
for all $row \in \mathbb{D}$.test **do**
$\mathbb{O} \leftarrow prediction(\mathbb{S}, \text{row})$
end for
return ($\mathbb{O}$)
End.

1. In this algorithm, the training data set is stored for prediction.

2. Next, it iterates over the test data and predict on it using our training data.

3. At the end, the algorithm produces the prediction in real-time.

4.2.4 Target Readings Selection Ω

Target Readings Selection is the prediction algorithm that predicts the future readings of each v in a sensor s. Lets discuss the target readings selection process with an example. Assume we are trying to lower the energy consumption in a house by using a data that has a set of

s and their v over time. It is also assumed that there are three sensor variables v_1^a, v_2^a, v_3^a in a house with readings that lead to energy consumption that belongs to class label a. Moreover, the algorithm suggested to lower the target consumption to the class that belongs to label b. For example, let's say that the total consumption at class a is 1300 measure unit, and we want to lower it to 1000 measure unit, which belongs to class b. Then, it is needed to figure out the optimal reading from class b that will allow us to reach the required total consumption. Hence, the selected reading tuple in class b is the tuple that satisfies the following equation:

$$Min(\sqrt{(v_1^b[i] - v_1^a)^2 + (v_2^b[i] - v_2^a)^2 + (v_3^b[i] - v_3^a)^2}) \tag{4.12}$$

where $V_k[i]$ is reading number i for variable k in the sensor.

Algorithm 6 represents the getting new values process and it operates as follows:

1. The algorithm will take the target consumption and the current consumption as inputs.
2. It will then iterate over the classes, and select the class that is the closest to the target consumption.
3. Afterwards, it will go through that specific class, and the variable values that will provide us with the minimum Euclidean distance.
4. The selected variable values will provide us with the new tuple, allowing the target consumption to be reached.

4.2.5 The Action selection process Υ

Action selection is a process that moves the sensor readings from one tuple to another to satisfy a certain objective constraint. This happens through selecting the new class label that will try

Algorithm 6 : Get New Values

Input : Target Consumption (TC), current .
Output : $(v_1^a, \ldots, v_k^a), (v_1^b, \ldots, v_k^b)$.

Begin
$Max \leftarrow 0$
$Selectedc \leftarrow \emptyset$
for all $c \in \mathbb{C}$ **do**
4: **if** c.Consumption ≤ TC and c.Consumption > Max **then**
$Max \leftarrow c.Consumption$
$Selectedc \leftarrow c$
end if
8: **end for**
$v_1^a \ldots v_k^a \leftarrow current.variables$
$min \leftarrow \infty$
for all $v_1^b \ldots v_k^b \in selectedc.variables$ **do**
12: $NT \leftarrow (\sqrt{(v_1^b - v_1^a)^2 + (v_2^b - v_2^a)^2 + (v_3^b - v_3^a)^2})$
if $min \geq NT$ **then**
$min \leftarrow NT$
$(v_1, \ldots, v_k) \leftarrow (v_1^b, \ldots, v_k^b)$
16: **end if**
end for
return $(v_1^a, \ldots, v_k^a), (v_1^b, \ldots, v_k^b)$
End.

to make adjustments to move closer to the new consumption target. For every variable, there is an action, and the choice of actions will achieve the change of classes that will take place.

Algorithm 7 represents the action selection process Υ and it operates as follows:

1. The algorithm will take the newly chosen tuple from the Algorithm 6 as an input.
2. Afterwards, the algorithm will iterate through all actions for every variable and checks if the action will move the variable to the desired value.
3. Lastly, for every variable, the suitable action that will allow us to reach the target variable values will be selected.

Algorithm 7 : Action Selection

Input : $(v_1^a, \ldots, v_k^a), (v_1^b, \ldots, v_k^b), \Lambda$.
Output : Variable Actions (VA).

Begin
$VA \leftarrow \emptyset$
for all $v_i^a \in (v_1^a, \ldots, v_k^a), v_i^b \in (v_1^b, \ldots, v_k^b)$ **do**
for all $\lambda \in \Lambda$ **do**
if $v_i^b \leftarrow \delta(v_i^a, \lambda)$ **then**
$VA \leftarrow VA \cup \lambda$
break
end if
end for
end for
return (VA)
End.

4.3 Modified Bayesian for Accumulative Learning

The proposed accumulative Bayesian learning algorithm updates the mean and the standard deviation of each class based on the new incoming data. The learning algorithm can be used in

various applications and it will be used in controlling the energy consumption. The motivation behind the algorithm involves starting with historical data in the beginning, learning the mean and standard deviation of each class via supervised learning. Afterwards, the new data items are classified into different classes and the mean and standard deviation of all classes are updated.

In the proposed algorithm, the curves and distributions are adjusted accumulatively. The more the data, the more the environment learns, and the higher the accuracy of the prediction gets. This is not the case with the non-learning normal Bayes, as the training sample is provided and the system functions without enhancements in a supervised learning approach. However, with the novel proposed algorithm, learning more useful behaviors will be constant, which means the more the data, the more accurate the predictions will be. The algorithm starts with a supervised learning approach at the beginning, then later, it keeps learning without the need for any supervision in an unsupervised learning approach, making the algorithm semi-supervised learning-based.

The algorithm is divided into two phases. In the first phase, the curve is enhanced by learning new data items based on the training data with labels, followed by classifying the new data items. Afterwards, the second phase takes place, which occurs during the run time after the supervised learning training is completed. During that run time, the class of the generated data is unknown. The probability of each new data item belonging to each class is calculated, where the highest probability indicates the class that the data items belongs to. Subsequently, the mean and standard deviation get updated, and the curve gets smoother when more data items are added.

This section is dedicated to demonstrate the inner workings of the algorithm.

4.3.1 Updating the Mean:

Since the mean is defined by:

$$\mu_n = \frac{\Sigma_{i=1}^{n} x_i}{n} \tag{4.13}$$

which also means that:

$$n\mu_n = \Sigma_{i=1}^{n} x_i \tag{4.14}$$

having a new value x_{n+1} makes the new μ as follows:

$$\mu_{n+1} = \frac{\Sigma_{i=1}^{n+1} x_i}{n+1} \tag{4.15}$$

therefore:

$$\mu_{n+1} = \frac{\Sigma_{i=1}^{n} x_i + x_{n+1}}{n+1} \tag{4.16}$$

From Equation 4.14 and Equation 4.15, it can be concluded that:

$$\mu_{n+1} = \frac{n\mu_n + x_{n+1}}{n+1} \tag{4.17}$$

$$\mu_{n+1} = \frac{n\mu_n + x_{n+1}}{n+1} \tag{4.18}$$

4.3.2 Updating the Standard Deviation:

Standard deviation of n elements is defined by:

$$\sigma_n = \sqrt{\frac{\Sigma_{i=1}^{n} (x_i - \mu_n)^2}{n}} \tag{4.19}$$

Standard deviation of $n+1$ elements is defined by:

$$\sigma_{n+1} = \sqrt{\frac{\Sigma_{i=1}^{n+1}(x_i - \mu_{n+1})^2}{n+1}} \tag{4.20}$$

which makes the variance of $n+1$ elements as:

$$\sigma_{n+1}^2 = \frac{\Sigma_{i=1}^{n+1}(x_i - \mu_{n+1})^2}{n+1} \tag{4.21}$$

This leads to:

$$\sigma_{n+1}^2 = \frac{\Sigma_{i=1}^{n}(x_i - \mu_{n+1})^2 + (x_{n+1} - \mu_{n+1})^2}{n+1} \tag{4.22}$$

$$\sigma_{n+1}^2 = \frac{\Sigma_{i=1}^{n}(x_i^2 - 2x_i\mu_{n+1} + \mu_{n+1}^2) + (x_{n+1} - \mu_{n+1})^2}{n+1} \tag{4.23}$$

$$\sigma_{n+1}^2 = \frac{\Sigma_{i=1}^{n}(x_i^2 - 2x_i\mu_{n+1} + \mu_{n+1}^2) + x_{n+1}^2 - 2x_{n+1}\mu_{n+1} + \mu_{n+1}^2}{n+1} \tag{4.24}$$

$$\sigma_{n+1}^2 = \frac{\Sigma_{i=1}^{n}x_i^2 - \Sigma_{i=1}^{n}2x_i\mu_{n+1} + \Sigma_{i=1}^{n}\mu_{n+1}^2 + x_{n+1}^2 - 2x_{n+1}\mu_{n+1} + \mu_{n+1}^2}{n+1} \tag{4.25}$$

$$\Sigma_{i=1}^{n}2x_i\mu_{n+1} = 2\mu_{n+1}\Sigma_{i=1}^{n}x_i = 2n\mu_n\mu_{n+1} \tag{4.26}$$

This gives the following formula:

$$\sigma_{n+1}^2 = \frac{\Sigma_{i=1}^{n}x_i^2 - 2n\mu_n\mu_{n+1} + \Sigma_{i=1}^{n}\mu_{n+1}^2 + x_{n+1}^2 - 2x_{n+1}\mu_{n+1} + \mu_{n+1}^2}{n+1} \tag{4.27}$$

$$\sigma_{n+1}^2 = \frac{\Sigma_{i=1}^{n} x_i^2 - 2n\mu_n\mu_{n+1} + (n+1)\mu_{n+1}^2 + x_{n+1}^2 - 2x_{n+1}\mu_{n+1}}{n+1} \tag{4.28}$$

which leads to:

$$\sigma_{n+1}^2 = \frac{\Sigma_{i=1}^{n+1} x_i^2 - 2n\mu_n\mu_{n+1} + (n+1)\mu_{n+1}^2 - 2x_{n+1}\mu_{n+1}}{n+1} \tag{4.29}$$

So the standard accumulative formula is:

$$\sigma_{n+1} = \sqrt{\frac{\Sigma_{i=1}^{n+1} x_i^2 - 2n\mu_n\mu_{n+1} + (n+1)\mu_{n+1}^2 - 2x_{n+1}\mu_{n+1}}{n+1}} \tag{4.30}$$

4.3.3 Classification algorithm

Unlike the original Bayesian's classifier, which is a supervised learning-based algorithm, the new proposed classifier is a semi-supervised learning-based approach. It starts as supervised learning at the beginning, then later, it keeps learning without the need for any supervision. Without the initial supervised learning step, classes could be separate for each row, since for every data element the mean is the value of the element and the standard deviation is zero. The classification is done in two phases, phase one is supervised when the class label is provided along with input data, and phase two is unsupervised when no label is provided.

Algorithm 8 describes phase one, and it operates as follows:

1. Initially, the algorithm takes the data and the labels as inputs.

2. Next, it iterates over each item in the data set, and check if the label of the data item belong to a certain class.

3. If the label does not belong to any class, then a class is created for that label.

4. After that, loop through every class item, and for every class column in the classes, the mean and standard deviation of that column is calculated.

5. At the end, all classes are returned as the output of the algorithm.

Phase 2 is the learning process during run time. This is done in two steps; the first step is the classification mechanism, and the second one is the distribution update. The classification is done based on the highest probability provided by a particular class.

Algorithm 9 describes phase two, and it operates as follows:

1. In this algorithm, the data and classes are taken as the inputs.

2. For every data item in the set, the probability for each class is calculated.

3. Whenever a new data item is added, check to see to which class that the data belongs to by calculating the probability.

4. The class with the highest probability are chosen.

5. After that, the mean and the standard deviation are updated by using Equations 4.18 and 4.30.

To explain classes in a better way, an example showing temperature classes will be discussed. First, the data is classified based on predefined labels. The classes are as follows:

1. Temperatures from -20°C and below belongs to class 1.

2. Temperatures between -19°C to -10°C belongs to class 2.

3. Temperatures between -9°C to 0°C belongs to class 3.

4. Temperatures between 1°C to 10°C belongs to class 4.

5. Temperatures between 11°C to 20°C belongs to class 5.

6. Temperatures between 21°C to 30°C belongs to class 6.

7. Temperatures from 30°C and above belongs to class 7.

At the beginning, classes are formed through supervised learning, where every data item will belong to a certain classification based on its label. After that, when a new unlabeled data item is added, the probability of this data item belonging to each class is calculated. Next, the highest probability indicates the class that the data items belong to. For example, if 14°C data item is added then the probability for this data item to belong to this class will be calculated and then the it will belong to the class that outputs the highest probability, where in this case it would be class 5. Each class will have its own mean and standard deviation, and they will keep updating whenever a new data item is added to that class. Hence, the classes are dynamic and due to this some class might merge avoiding over classification issues. This modification takes place whenever two classes start having very similar mean and standard deviation after they keep updating, therefore ending up as one class.

Algorithm 8 : Phase 1: Supervised Learning

Input : data $\mathbb{D}$, labels $\mathbb{L}$.
Output : classes $\mathbb{C}$

Begin
$\mathbb{C} \leftarrow \emptyset$
$index \leftarrow 0$
for all $d \in \mathbb{D}$ **do**
  if $\mathbb{L}(index) \notin \mathbb{C}$ **then**
    $c \leftarrow$ newClass()
    $c.label \leftarrow \mathbb{L}(index)$
  else
    $c \leftarrow$getClass($\mathbb{L}(index)$)
  end if
  $c \leftarrow c \cup d$
end for
for all $c \in \mathbb{C}$ **do**
  $index \leftarrow 0$
  for all $col \in \mathbb{C}.Columns$ **do**
    $c.\mu(index) \leftarrow$ mean($c.column(index)$)
    $c.\sigma(index) \leftarrow$ STDEV($c.column(index)$)
    $index \leftarrow index + 1$
  end for
  $c.Length(i) \leftarrow index$
end for
return ($\mathbb{C}$)
End.

Algorithm 9 : Phase 2: Run Time Learning

Input : data d , classes $\mathbb{C}$.
Output : classes $\mathbb{C}$
Begin
$index \leftarrow 0$
$\mathbb{P}_{max} \leftarrow 0$
$\mathbb{C}_{max} \leftarrow \emptyset$
for all $col \in d$ **do**
  $\mathbb{P} \leftarrow 1$
  for all $c \in \mathbb{C}$ **do**
    $\mathbb{P} \leftarrow \mathbb{P} \times NormalDistribution(col, c.\mu[index], c.\sigma[index])$
  end for
  if $\mathbb{P}_{max} < \mathbb{P}$ **then**
    $\mathbb{P}_{max} \leftarrow \mathbb{P}$
    $\mathbb{C}_{max} \leftarrow c$
  end if
  $index \leftarrow index + 1$
end for
$\mathbb{C}_{max} \leftarrow \mathbb{C}_{max} \cup d$
update μ according to equation 4.18 for every column
update σ according to equation 4.30 for every column
return ($\mathbb{C}$)
End.

Chapter 5

Simulation and Experiments

Energy consumption management is one of the major challenges faced in a smart city, due to the lack of resources and the rapid growth of the world's population. In particular, governments have been trying to tackle the energy over-consumption issues by implementing different policies such as "the more you consume, the more you pay". While such policies help reduce energy consumption, it does not guarantee that the consumption does not exceed a certain target thresholds. It is becoming ever so important for smart cities to try to set maximum thresholds to avoid blackouts that result from over-consumption.

For example, countries in the Gulf Cooperation Council (GCC) area, where the temperature reaches above 50 degree Celsius, have high possibilities of blackouts and electricity grid cut offs, particularly in the summer when people over-consume energy due to the overuse of air-conditioners and dehumidifiers. One potential solution includes setting energy consumption limit thresholds for each house across the smart city, in order to try to manage the energy consumption and ensure the electrical grid can keep providing what is necessary for the city to function.

Implementing such a policy or strategy raises many issues in governance, privacy, individual

rights, etc.. This is used as a general scenario to illustrate our model and to illustrate the approach and algorithms. Such a scenario is not entirely hypothetical since a city would have control of the buildings it manages and would be able to introduce operational policies to reduce or manage energy consumption in its own buildings. For our experiments, they are described in terms of "houses".

One question that does arise is the maximum threshold boundary that needs to be set for each house or building. This is a probabilistic classification problem since every house might fall into different categories. It is worth noting that the consumption of every house is independent of the consumption of other houses and so there is no "global" control. Since the problem is probabilistic and houses are independent, NB classifier is a an excellent candidate to be used as a solution since it assumes that variables are independent. Bayesian classification is simple and fast which makes it very suitable for real-time systems, which is the case in energy grid control systems.

In our overarching scenario of energy management, it is assumed that the system depends on a sensor network that collects readings for decision-making. Every sensor has one or more variable(s). Every variable reads some phenomena like temperature or humidity. Every sensor is associated with an intelligent agent that takes the decisions; such agents may run in any layers of our computational hierarchy, though most likely they would run in the IoT or Edge layers. Decisions made by such agents are based on a local objective that is defined for the particular house where the sensor operates. Agents that control sensors try to meet the objectives assigned to them in each house. Their assigned objective is not to exceed the total consumption energy calculated in kilowatt-hours (kWhs). The agent will try to reduce the consumption by controlling settings of sensors in order to reach this level. Every agent maintains a history and based on that history it tries to reach a certain consumption level. The more the agent learns, the better and more accurate the settings' selections are (the actions). The agent selects the action by altering the settings, such actions would be choosing the right temperature for an air

conditioner or the humidity level for a humidifier.

The local objectives is calculated by determining the number of people living in each house, and are uniquely assigned to every house based on the level of over-consumption. This is also done by classifying houses into categories based on their consumption. The global energy consumption control process starts by tackling the highest consuming houses, then it moves to the categories that are consuming less energy. The classification of houses is determined by learning the consumption distribution, and then determining where each house is exactly on the distribution. Once the supervised learning is completed, the classification will take place in real-time. If the consumption of a certain house is more than the mean plus half of the standard deviation, then it is considered a house with high consumption levels. If it consumes more than the mean, but lower than the mean plus half of the standard deviation, then it is considered above average.

Note that classification is vital since it provides important information for categorizing houses based on their consumption for decision-making purposes. The algorithms learn which settings of devices, such as air conditioners, humidifiers, heaters, and dehumidifiers, lead to a particular consumption class. Then, when it is required to move a certain house from one consumption level to another, the settings that are highly likely to lead to a certain consumption are chosen. Note that classes could have different lengths. The class is defined through supervised learning process when feeding the system with a certain input and the label associated to it. The input is basically the current readings and the target readings. Values that have the same consumption range get the same label. That range is determined by the designer or administrator. In this thesis, it is assumed that the reading range takes place every 10 measurements; a single reading could belong to multiple classes with different probabilities. Hence, the class that gives the highest probability is the class of this reading. If the following row of values are given:

1. Current temperature (T_c)

2. Target temperature (T_t)

3. Current humidity (H_c)

4. Target humidity (H_t)

5. Consumption (c)

Then this row belong to class $\mathbb{C}$ if the following is the maximum among all different classes, where $\mathbb{P}(x|y)$ is the probability of x given y:

$$\mathbb{P}(T_c|\mathbb{C}) \times \mathbb{P}(T_t|\mathbb{C}) \times \mathbb{P}(H_c|\mathbb{C}) \times \mathbb{P}(H_t|\mathbb{C}) \times \mathbb{P}(c|\mathbb{C})$$

It is worth noting that labels are just unique class names that are assigned as identifiers for every class.

In this chapter, the different simulations are described and explained, assuming the overarching scenario described above and using the model described in Chapter 4. Also, it is followed by presenting and discussing the results of the experiments. The simulation experiments are aimed to minimize the energy consumption in a city for a given number of houses. The modified Bayesian for accumulative learning was applied to the experiments. The way that the Bayesian algorithm works is by learning from the historical data, where the mean and standard deviation are learned. The classification into different classes at the beginning is done by supervised learning. Moreover, whenever a new data is added, we check to see to which class that data belongs. When deciding the classification of the new data, the mean and standard deviation gets updated. The temperature and humidity data used is taken from The Weather Network [74], on which the distribution is built. The data describes the weather in Toronto, Ontario, and the data taken spans from the last week of November 2020 and until the first week of December 2020, accumulating two weeks of data.

5.1 Simulator

The simulator was programmed using Java version 8, where 30 classes were implemented in the simulator framework and each class represented a particular functionality. The simulator emulated the physical layer model described in Chapter 4. The description of the classes are explained below:

Total Consumption: In this class, the total energy consumption is calculated by checking the total time each electrical device is turned on then off. The time difference between the time the device is turned on and the time the device turns off is taken then multiplied by the consumption rate of the device. The energy consumption is calculated in kilowatt-hours (kWhs). This technique is replicated from the paper of Karnouskos, *et al.* [75].

Air Conditioner: This class inherits from the Total Consumption class and the consumption rate per second of an air conditioner is recorded. The air conditioner is set to have 0.25 watts per second per room [75].

Heater: This class inherits from the Total Consumption class and the consumption rate per second of the heater is noted. The heater is set to have 1.50 watt per second per room [75].

Humidifier: This class inherits from the Total Consumption class and the consumption rate per second of the humidifier is noted. The humidifier is set to have 0.25 watt per second per room [75].

Lights: This class inherits from the Total Consumption class and the consumption rate per second of the lights is noted. The lights is set to have 0.0112 watt per second per room [75].

Normal Distribution: Normal distribution is an extremely important continuous probability distribution in statistics. It has the ability to describe how the values of a variable are distributed. In our simulator, the normal distribution is implemented in a class that provides both

the mean and standard deviation, which belongs to the distribution or relative to the range defined by maximum and minimum boundaries.

Event: This class is added as event-based simulation is a simple yet versatile way of describing a dynamic system. Simulations rely on a model of a real-world process to imitate time-dependent behavior.

Event List: The simulation maintains at least one list of simulation events. This is sometimes called the pending event set because it lists events that are pending as a result of previously simulated event but have yet to be simulated themselves.

Simulation Clock: When the simulation of a model is related to time, it will obtain the simulated time from the clock of the simulation. The amount of time spent on simulating a model is the simulation time generated, which can be broken down into years, months, days, hours, and minutes.

Smart City Object: A smart city object includes the event list, simulation clock, kilowatt consumption, and the target reading. So that any class representing an object in a smart city, such as sensor, room, house, etc. that inherits from this class can share the same information and settings.

Sensor Variables: In this class, the sensor variables are implemented as described in the model previously. Every sensor might have one or more variables. Every sensor variable has a history, and we keep track of the sequence of the variable readings. Moreover, each sensor variable has an ID, type, and a value. The duration and the maximum and minimum value of each sensor variable is recorded to be able to increase or decrease on the reading.

Sensor: The sensor class inherits from the Smart City Object class and it has an array of sensors variables. Each sensor has an ID, and the ability to add, increase or decrease the number of sensor variables for every sensor.

Room: The room class extends from Smart City Object and it has an array of sensors. Each room has an ID, air conditioner, heater, lights, and a humidifier. As well as, the ability to add, increase or decrease the number of sensors in every room.

House: The house class extends from Smart City Object and it has an array of rooms. Each house has an ID, and the ability to add, increase or decrease the number of rooms. Every house has a random number of rooms.

City: The city class inherits from the Smart City Object and it has an array of houses. The city has an ID, and the ability to add, increase or decrease the number of houses. A city has a random number of houses that is provided by the user.

Simulation Time Stamp: This class provides a time stamp for each entry. The stamp includes the year, month, day, hour, minute and second for each entry at the time it takes place.

Reading: This class will provide a time stamp for each sensor's variable reading.

Weather Network Entry: In this class, the maximum and minimum number are stored for each data entry. Also, the mean and standard deviation is calculated for each one. The mean and the standard deviation are calculated using the maximum and minimum values. As shown in Figure 5.1, the range of the distribution is from -3 σ to 3 σ, which means that the total length is 6σ. Hence, it can be said that:

$$6\sigma = (max - min) \tag{5.1}$$

where,

$$\sigma = \frac{max - min}{6}. \tag{5.2}$$

The mean is calculated using:

$$\mu = \frac{max - min}{2} \tag{5.3}$$

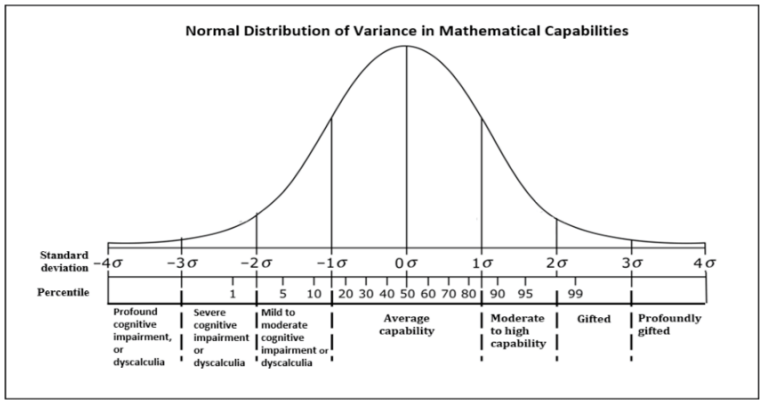


Figure 5.1: Variance of Normal Distribution

Weather Network Statistics: In this class, the set of information is defined and classified. As we are analyzing day-to-day activities, it is mandatory to have an overview of the data sets. Accordingly, the time of sunrise and sunset has been added to have better results. The maximum and minimum temperature and humidity are included as well for each month to be able to analyze the sensors.

Data Row: A row is basically a set of values. Those values are the readings of sensor variables as well as some aggregates like mean, standard deviation, length, sum and so on. Every row in the data set is added into a list, and the ability to add to the list and get any specific values from the list.

Data Column: Data column is a set of data rows. In this class, the mean and standard deviation for each column is calculated. The incremental Bayesian learning that was introduced by us in section 4.3 is implemented here, where when any value is added to a class, the mean and standard deviation are updated by using Equations 4.18 and 4.30.

Data Table: Data table allows us to add columns to the rows that exists. The class helps us to keep adding column to each row and to get the the index of a specific column if it is needed to get some statistics or aggregates out of this column.

Data Reader: This class reads the Weather Network data files [74] and place it into a table

using the previous class Data Table that was discussed.

Bayesian Classifier: The Bayesian Classifier class reads the table and trains on the data. It goes though all the data and classifies it into classes.

Classes: In this class, the label is generated for each data table. It will check if a class has a label, and if a class do not have a label yet, a label will be created.

Local Objective: Local Objective includes the variable name, the target value, the weight, the cost per second, and the time for unit change. The target value is the value we are trying to reach. The weight is the cost weight being passed for each variable. The cost per second is the consumption rate for each variable. Furthermore, the time for unit change is the duration it takes for each unit to change. The assumption made is that the temperature needs 10 minutes to go up or down by 1 degree, and the humidity required 1 minute to go up or down by 1%. The time for unit change can be altered by the user and other assumptions can be made.

Local Objectives: Local Objectives class takes a list of Local Objective. A local objective can be added to the list as well.

Experiment Manager: This class controls the experiments that will be implemented. As each experiment will have different objectives, in this class the settings of each experiment will be illustrated and defined.

Smart City: This is the controller of the simulator.

Figure 5.2 represents the class diagram for some major classes in the simulator. A full class diagram and a more detailed one can be found in the Appendix in Figure A.1

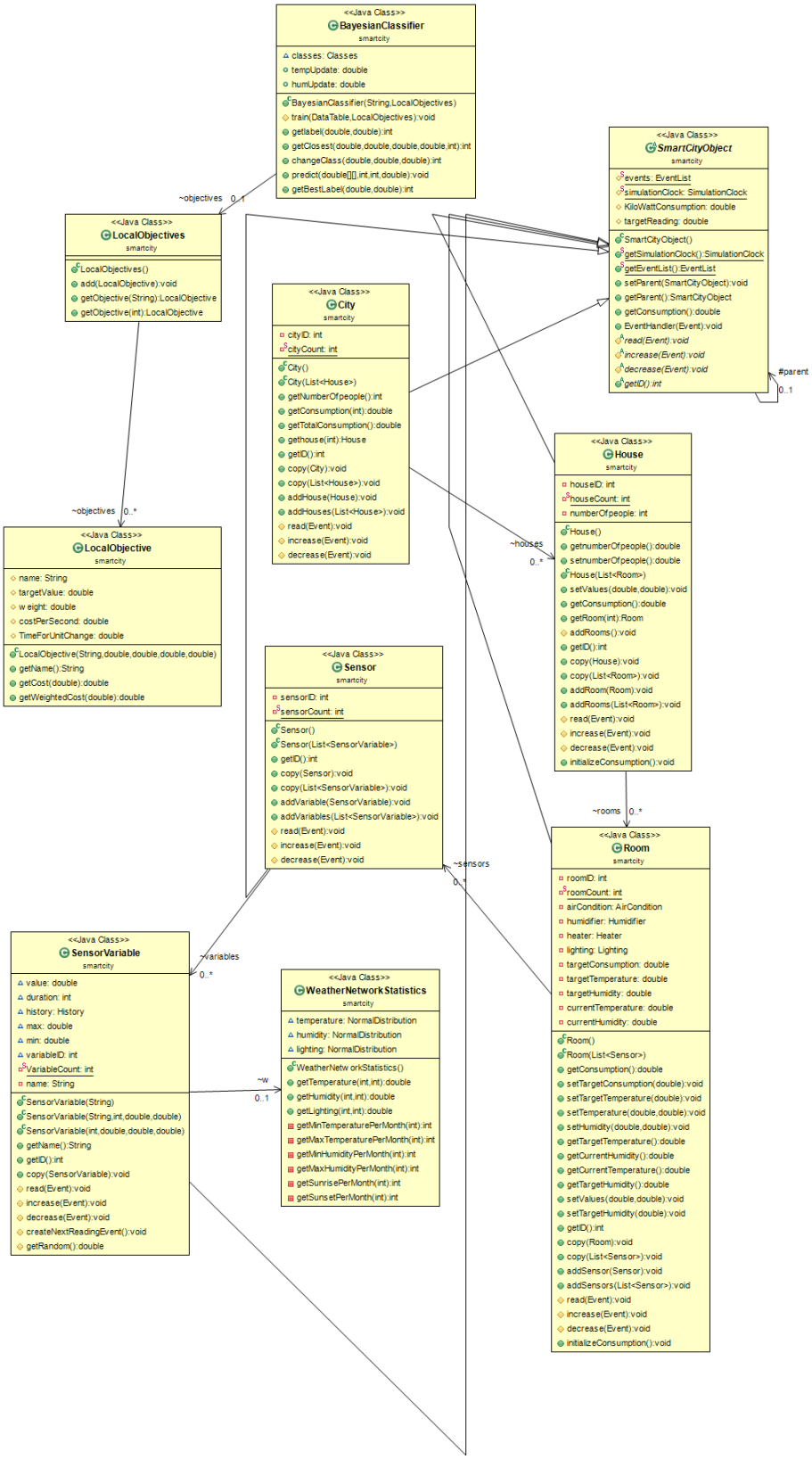


Figure 5.2: Simulator Class Diagram

5.2 Experiments

In this section, the experiments are presented, where the focus will be driven towards the problem of addressing the minimizing of energy across a city. Several experiments with different scenarios are presented along with their results. For all experiments, it is assumed that there is a city that has 100-1000 houses, where every house has a random number of rooms that can range from 3 rooms, and up to 15 rooms. In every house there will be at least one sensor that has three variables, which are the temperature, humidity and motion sensors. Every house has its own consumption based on the amount of energy used per house. The total city consumption is calculated per hour, where the whole city has a global objective. Moreover, the idea is to reduce the overall consumption only when it is more than the assumed maximum capacity.

The Total Consumption class in the simulator will always calculate the sum of the consumption of all the houses in the city per hour. Based on that, it will choose the houses that exceeded the the consumption rate and violated the local objective, and try to reduce their consumption to meet the global objective. This will be done by the agent which tries to predict on the target readings selection, and select the suitable actions to lower the consumption. Also, by taking into consideration the number of rooms in the houses, and the number of individuals who live in the house. Since, as the number of rooms and individuals in a certain house increases, this will lead to the increase of the consumption as more energy will be utilized.

The number of individuals in each house will be generated randomly. It is expected that each house will have 0 occupants and up to 6 occupants on average. Based on the number of rooms and individuals in each house, the objective will be set. The house shouldn't exceed that limit to avoid exceeding the global objective. When a decision is taken, the sensors start reading so if people change rooms or leave their houses, it will be recognized in the next run.

5.2.1 Data

From Figure 5.3, it can be seen that the temperature and humidity sensor variables' values are plotted since they affect the total consumption of the city. Hence, based on the temperature, the heating or air conditioner will be turned on or off, and based on the humidity, the humidifier or dehumidifier will be turned on or off. Accordingly, these electrical devices will affect the kilowatts-hour (kWhs) that we are trying to compute. The lighting of the rooms will be controlled by checking whether individuals are inside or outside the room, and depending on the time of the day.

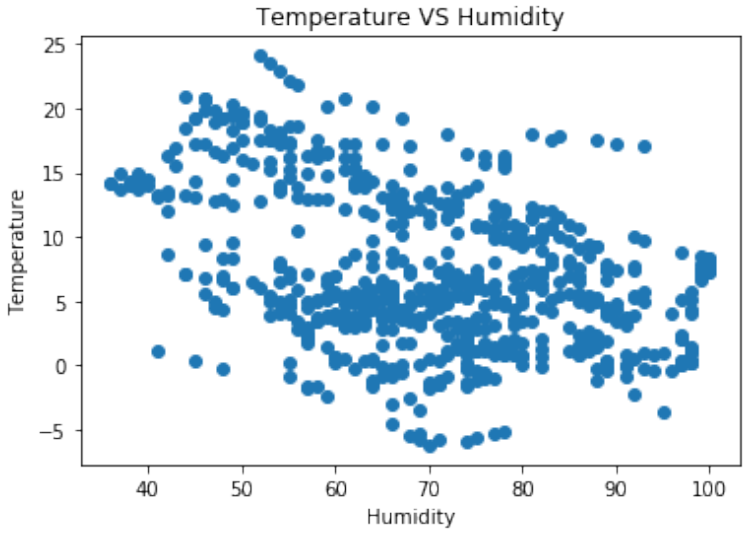
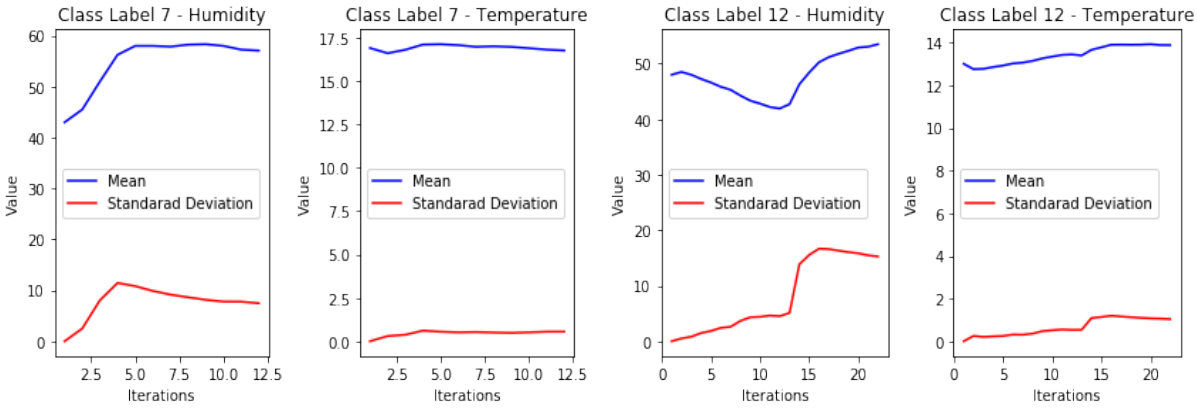


Figure 5.3: Temperature VS Humidity

So, the temperature and humidity are controlled in these experiments and the variation of values of these variables is dependent on the external temperature and that is why the data on external temperature and humidity was extracted from The Weather Network [74]. Figure 5.3 is the pre-processed data that is used for temperature and humidity, and moving forward this data will be classified into different classes.

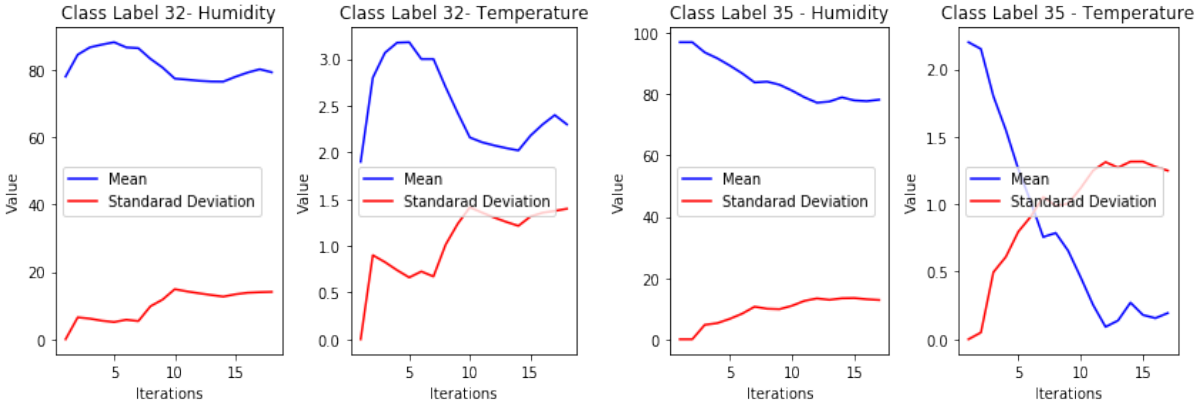
After classifying the data, the labels and classes are summarized in Tables 5.1 and 5.2. The tables include the class label, the name of the variable, the mean, the standard deviation, and

the length of each class. Figures 5.4 and 5.5 illustrates how the classes learn the mean and the standard deviation. The learning take place whenever a new data item is added to a class. As can be seen, the curves illustrate that learning occurs while data is collected and training takes place. The training data is limited and accordingly the learning can certainly be improved, if the training data was larger. In the second phase during the run time, the learning will move from supervised learning to unsupervised learning.



(a) Class with Label 7 **(b)** Class with Label 12

Figure 5.4: Classes with Labels 7 and 12



(a) Class with Label 32 **(b)** Class with Label 35

Figure 5.5: Classes with Labels 32 and 35

In the following subsections, the experiments and their results are explained. Each experiment constraints will be demonstrated and the results will be discussed. Every experiment is repeated and goes through 30 iterations, and the output is illustrated and summarized using graphs.

Table 5.1: Classes Statistics for Humidity and Temperature - Part 1

Label	Name	Mean	STD	Length
1	Humidity	48.85	5.30	7
1	Temperature	20.53	0.38	7
2	Humidity	51.10	4.06	10
2	Temperature	20.11	1.00	10
3	Humidity	50.56	5.54	9
3	Temperature	19.64	1.23	9
4	Humidity	52.0	3.60	8
4	Temperature	19.79	2.35	8
5	Humidity	54.83	5.58	6
5	Temperature	17.87	0.60	6
6	Humidity	51.56	4.52	9
6	Temperature	17.20	0.24	9
7	Humidity	57.08	7.47	12
7	Temperature	16.76	0.55	12
8	Humidity	55.63	7.24	8
8	Temperature	16.16	0.51	8
9	Humidity	62.33	15.40	6
9	Temperature	16.22	1.31	6
10	Humidity	61.43	14.59	14
10	Temperature	15.56	1.00	14
11	Humidity	56.94	16.18	17
11	Temperature	14.75	1.00	17
12	Humidity	53.50	15.26	22
12	Temperature	13.88	1.04	22
13	Humidity	53.83	9.63	6
13	Temperature	13.10	0.16	6
14	Humidity	66.29	7.67	14
14	Temperature	13.07	0.53	14
15	Humidity	68.25	3.19	8
15	Temperature	12.46	0.36	8
16	Humidity	68.90	5.52	10
16	Temperature	11.94	0.56	10
17	Humidity	70.50	8.51	10
17	Temperature	11.36	0.77	10
18	Humidity	73.00	12.13	7
18	Temperature	11.12	0.94	7
19	Humidity	69.13	13.47	15
19	Temperature	10.11	1.14	15
20	Humidity	73.64	10.90	14
20	Temperature	9.65	1.10	14
21	Humidity	65.91	14.30	22
21	Temperature	8.40	1.29	22
22	Humidity	66.80	14.38	15
22	Temperature	7.68	1.38	15

Table 5.2: Classes Statistics for Humidity and Temperature - Part 2

Label	Name	Mean	STD	Length
23	Humidity	71.00	13.07	17
23	Temperature	7.45	1.30	17
24	Humidity	67.19	10.21	26
24	Temperature	20.11	1.00	26
25	Humidity	67.90	10.846	48
25	Temperature	5.83	1.07	48
26	Humidity	78.46	12.39	35
26	Temperature	6.23	1.21	35
27	Humidity	72.25	14.50	40
27	Temperature	4.98	1.45	40
28	Humidity	75.36	13.63	39
28	Temperature	4.57	1.38	39
29	Humidity	77.05	10.23	21
29	Temperature	4.03	1.01	21
30	Humidity	80.89	11.46	17
30	Temperature	3.75	1.16	17
31	Humidity	71.53	14.80	19
31	Temperature	2.33	1.35	19
32	Humidity	79.28	14.07	18
32	Temperature	2.30	1.40	18
33	Humidity	74.30	7.70	27
33	Temperature	1.14	0.83	27
34	Humidity	77.77	9.11	22
34	Temperature	0.73	0.93	22
35	Humidity	78.24	12.83	17
35	Temperature	0.19	1.25	17
36	Humidity	79.05	11.11	20
36	Temperature	-0.45	1.08	20
37	Humidity	84.86	9.77	14
37	Temperature	-0.48	1.01	14
38	Humidity	94.00	7.76	16
38	Temperature	-0.20	0.83	16
39	Humidity	82.50	13.50	2
39	Temperature	-1.90	1.50	2
40	Humidity	66.00	0.00	1
40	Temperature	-4.50	0.00	1
41	Humidity	92.00	0.00	1
41	Temperature	-2.20	0.00	1
42	Humidity	68.67	0.47	3
42	Temperature	-5.50	0.16	3
43	Humidity	77.14	7.77	7
43	Temperature	-5.37	0.79	7

5.2.2 Experiment 1

The first experiment consists of 100 houses, where there is a random number of rooms and a random number of people in each house. The following are the assumptions for the experiment:

1. There is one sensor in each house and the sensor has one or more variables.

2. Each house has its own local objective based on the number of people and the relative energy consumption with respect to the mean, and the target consumption. Figure 5.6 illustrates houses with different targets (local objectives) that they are trying to achieve.

```
house 1 has high consumption. It is consuming 4485.0 target is 4036.5 target class 40
house 3 has high consumption. It is consuming 3945.0 target is 3550.5 target class 35
house 4 has high consumption. It is consuming 4815.0 target is 4333.5 target class 43
house 5 has high consumption. It is consuming 4350.0 target is 3915.0 target class 39
house 9 has high consumption. It is consuming 3900.0 target is 3510.0 target class 35
house 15 has high consumption. It is consuming 3705.0 target is 3334.5 target class 33
house 17 has high consumption. It is consuming 4620.0 target is 4158.0 target class 41
house 21 has high consumption. It is consuming 3480.0 target is 3132.0 target class 31
house 23 has high consumption. It is consuming 3465.0 target is 3118.5 target class 31
house 24 has high consumption. It is consuming 2925.0 target is 2632.5 target class 26
house 26 has high consumption. It is consuming 3840.0 target is 3456.0 target class 34
house 27 has high consumption. It is consuming 4965.0 target is 4468.5 target class 44
house 30 has high consumption. It is consuming 3015.0 target is 2713.5 target class 27
```

Figure 5.6: Local Objectives of Houses

3. The global objective the city is trying to achieve is to not exceed a threshold limit of 90% of the original consumption which is 155096 kilowatts. This will reduce the energy consumption by at least 10%.

4. Global objective is changed to have a threshold limit of 85% of the original consumption when the experiment started. This will reduce the energy consumption by at least 15%.

5. Global objective is changed to have a threshold limit of 75% of the original consumption when the experiment started. This will reduce the energy consumption by at least 25%.

Results for Experiment 1

The results for the experiment with the different global objectives are illustrated in Figures 5.7, 5.8, and 5.9. As can be seen, all the consumption are below the threshold, which validates our experiment. In Figure 5.9 the energy consumption percentage was very close to the threshold but it did not exceed it; it remains below 1% of the threshold and did not exceed it.

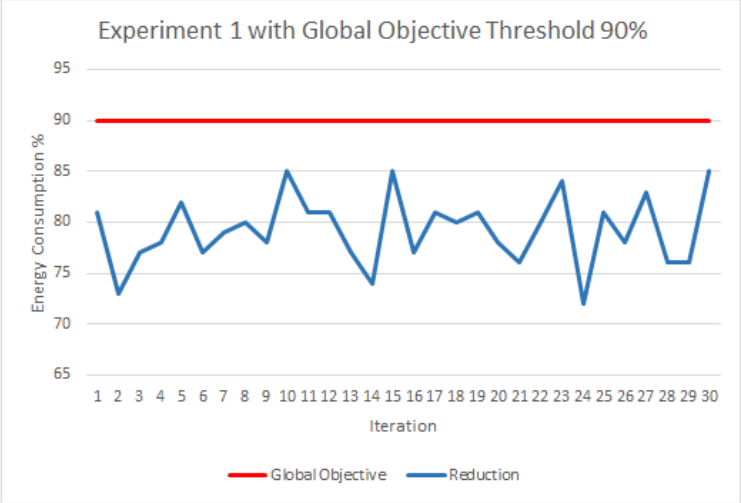


Figure 5.7: Experiment 1 - 90% Global Objective

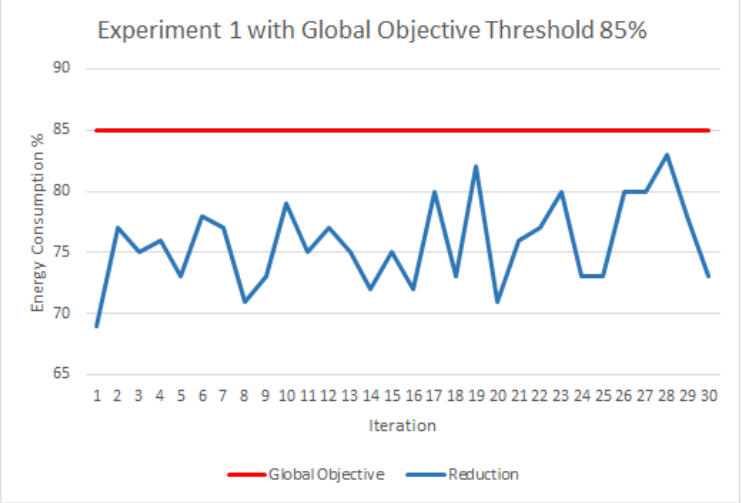


Figure 5.8: Experiment 1 - 85% Global Objective

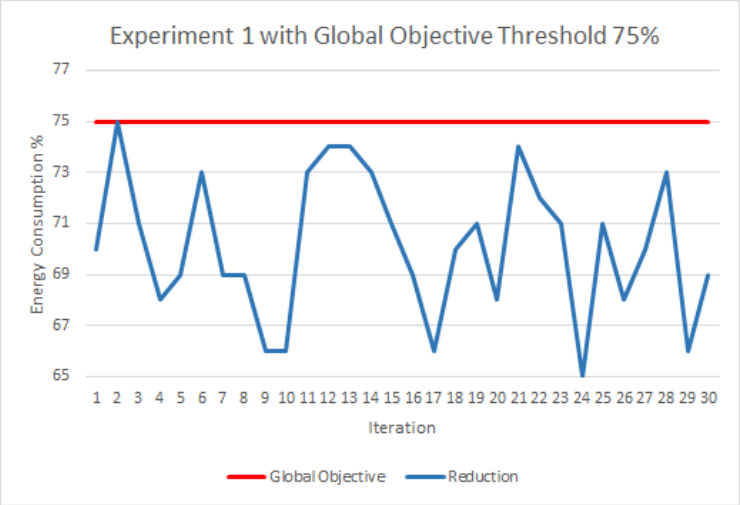


Figure 5.9: Experiment 1 - 75% Global Objective

5.2.3 Experiment 2

The second experiment is similar to the first experiment but uses 500 houses with random numbers of rooms and people instead. The following are the assumptions for the experiment:

1. There is one sensor in each house and the sensor has one or more variables.

2. Each house has its own local objective based on the number of people and the relative energy consumption with respect to the mean, and the desired target consumption.

3. The global objective the city is trying to achieve is to not exceed a threshold limit of 90% of the original consumption which is 772407 kilowatts. This will reduce the energy consumption by at least 10%.

4. Global objective is changed to have a threshold limit of 85% of the original consumption when the experiment started. This will reduce the energy consumption by at least 15%.

5. Global objective is changed to have a threshold limit of 75% of the original consumption when the experiment started. This will reduce the energy consumption by at least 25%.

Results for Experiment 2

The results for the experiment with different global objectives are illustrated in Figures 5.10, 5.11, and 5.12. As can be seen, all the consumption levels are below the threshold, which validates this experiment as well. This indicates that the model is able to control a city with an increased number of houses.

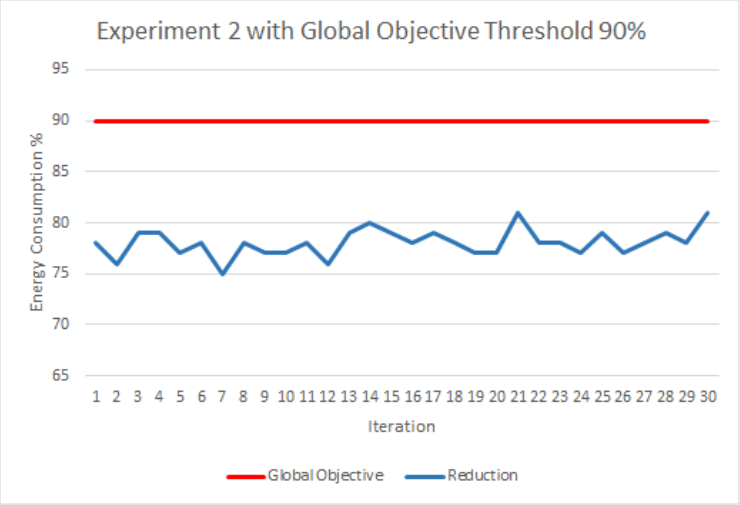


Figure 5.10: Experiment 2 - 90% Global Objective

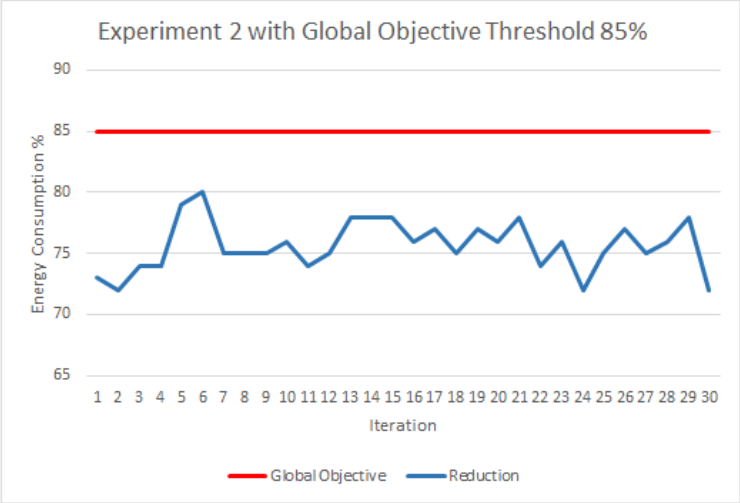


Figure 5.11: Experiment 2 - 85% Global Objective

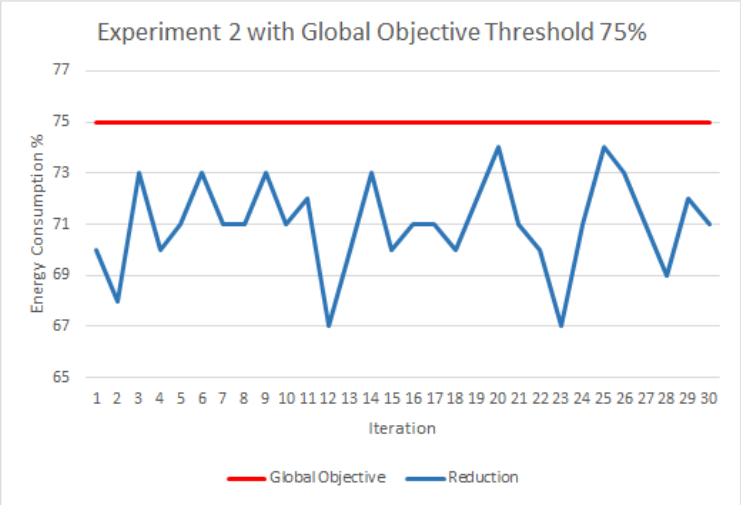


Figure 5.12: Experiment 2 - 75% Global Objective

5.2.4 Experiment 3

The third experiment is implementing a larger scale environment, where 1000 houses are used, and there is a random number of rooms and a random number of people in each house. The following are the assumptions for the experiment:

1. There is one sensor in each house and the sensor has one or more variables.
2. Each house has its own local objective based on the number of people and the relative energy consumption with respect to the mean, and the desired target consumption.
3. The global objective the city is trying to achieve is to not exceed a threshold limit of 90% of the original consumption which is 1450987 kilowatts. This will reduce the energy consumption by at least 10%.
4. Global objective is changed to have a threshold limit of 85% of the original consumption when the experiment started. This will reduce the energy consumption by at least 15%.
5. Global objective is changed to have a threshold limit of 75% of the original consumption when the experiment started. This will reduce the energy consumption by at least 25%.

Results for Experiment 3

The results for Experiment 3 with different global objectives are illustrated in Figures 5.13, 5.14, and 5.15. As can be seen, the consumption levels are below the threshold when the global objective had a threshold limits of 90% and 85%, whereas when the global objective had a threshold limit of 75%, the consumption levels exceed the limit in some iterations. The consumption levels did not exceed the limit by a big difference and this is expected at some point of times as the system keeps trying to achieve the objective but it is not always guaranteed that it will be fully met. Even though the number of houses has doubled the number in Experiment 2, the model is generating the desired output, and able to control the consumption.

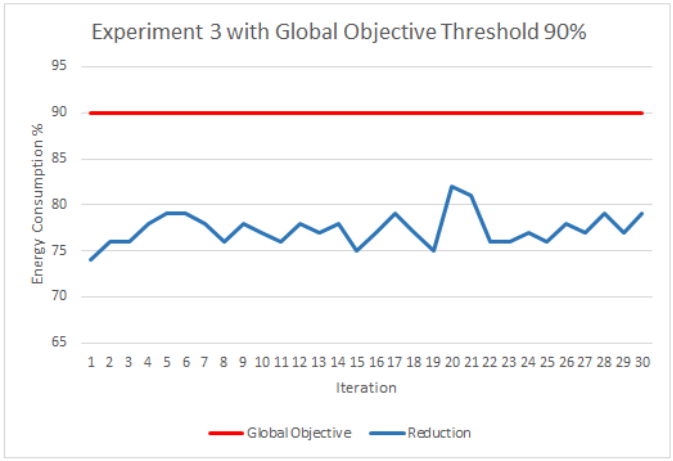


Figure 5.13: Experiment 3 - 90% Global Objective

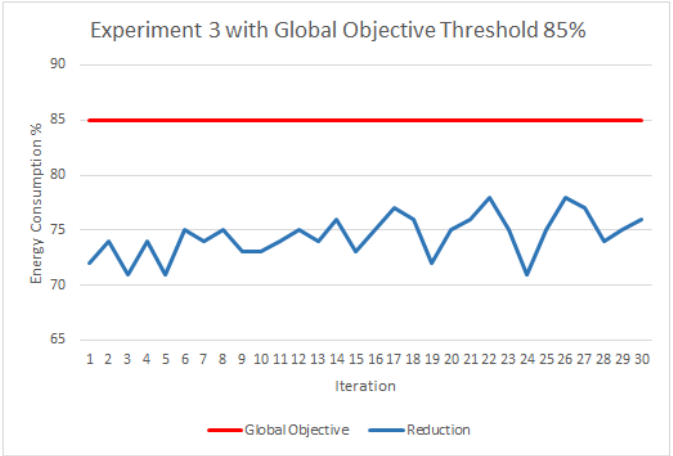


Figure 5.14: Experiment 3 - 85% Global Objective

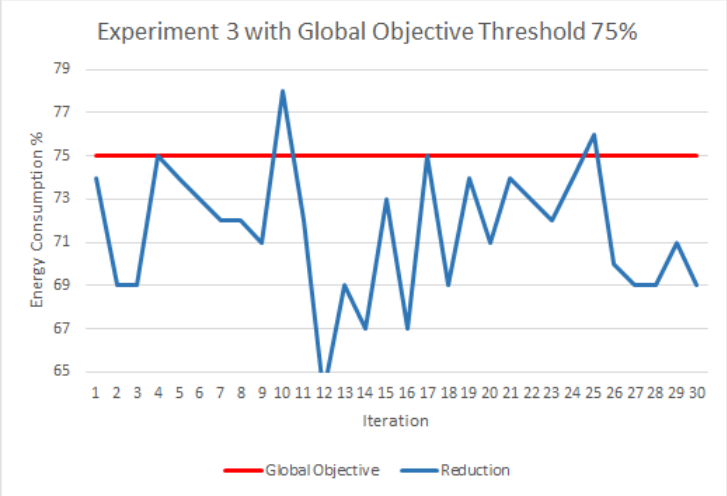


Figure 5.15: Experiment 3 - 75% Global Objective

5.2.5 Experiment 4

The fourth experiment consists of 100 houses, where there is a random number of rooms and a random number of people in each house. The following are the assumptions for the experiment:

1. There are multiple rooms in each house. Each room has a sensor, and every sensor has one or more variables. This means that a house has multiple sensors.

2. Each house has its own local objective based on the number of people and the relative energy consumption with respect to the mean, and the desired target consumption.

3. The global objective the city is trying to achieve is to not exceed a threshold limit of 90% of the original consumption which is 209415 kilowatts. This will reduce the energy consumption by at least 10%.

4. Global objective is changed to have a threshold limit of 85% of the original consumption when the experiment started. This will reduce the energy consumption by at least 15%.

5. Global objective is changed to have a threshold limit of 75% of the original consumption when the experiment started. This will reduce the energy consumption by at least 25%.

Results for Experiment 4

The results for Experiment 4 with different global objectives are illustrated in Figures 5.16, 5.17, and 5.18. In this experiment and the upcoming two, there are multiple sensors in each house, which is expansion from the first three experiment. From Figures 5.16, 5.17, and 5.18, it can be noticed that energy consummations percentage are below the threshold, which verifies this experiment as well.

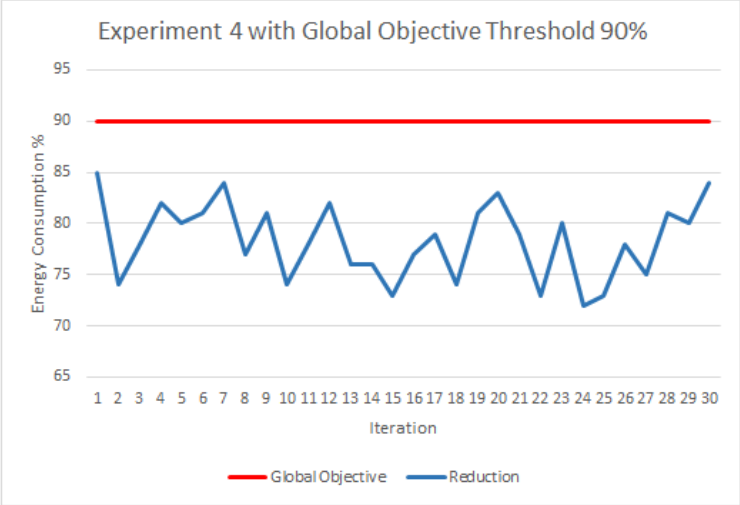


Figure 5.16: Experiment 4 - 90% Global Objective

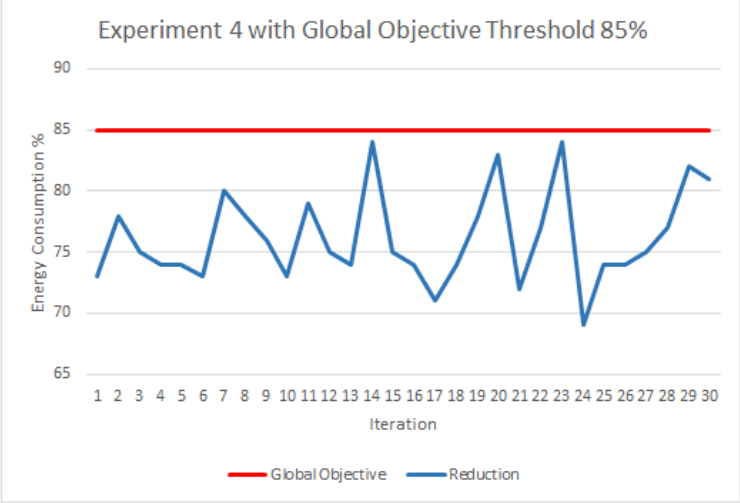


Figure 5.17: Experiment 4 - 85% Global Objective

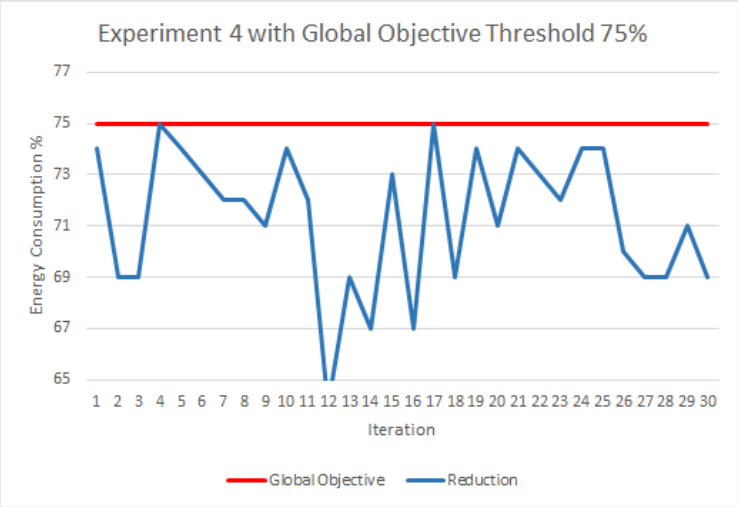


Figure 5.18: Experiment 4 - 75% Global Objective

5.2.6 Experiment 5

The fifth experiment is using 500 houses, where there is a random number of rooms and a random number of people in each house. The following are the assumptions for the experiment:

1. There are multiple rooms in each house. Each room has a sensor, and every sensor has one or more variables. This means that a house has multiple sensors.

2. Each house has its own local objective based on the number of people and the relative energy consumption with respect to the mean, and the desired target consumption.

3. The global objective the city is trying to achieve is to not exceed a threshold limit of 90% of the original consumption which is 1196130 kilowatts. This will reduce the energy consumption by at least 10%.

4. Global objective is changed to have a threshold limit of 85% of the original consumption when the experiment started. This will reduce the energy consumption by at least 15%.

5. Global objective is changed to have a threshold limit of 75% of the original consumption when the experiment started. This will reduce the energy consumption by at least 25%.

Results for Experiment 5

The results for Experiment 5 with different global objectives are illustrated in Figures 5.19, 5.20, and 5.21. From Figures 5.19, 5.20, and 5.21, it can be noticed that energy consummations percentage are below the threshold, which verifies this experiment as well. The number of houses has been increased by 5 times compared to Experiment 4, and the model is generating the desired output, and able to control the consumption regardless.

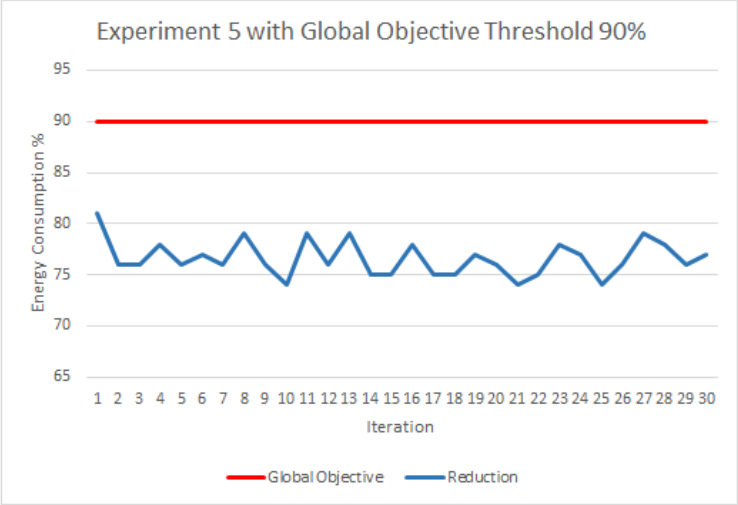


Figure 5.19: Experiment 5 - 90% Global Objective

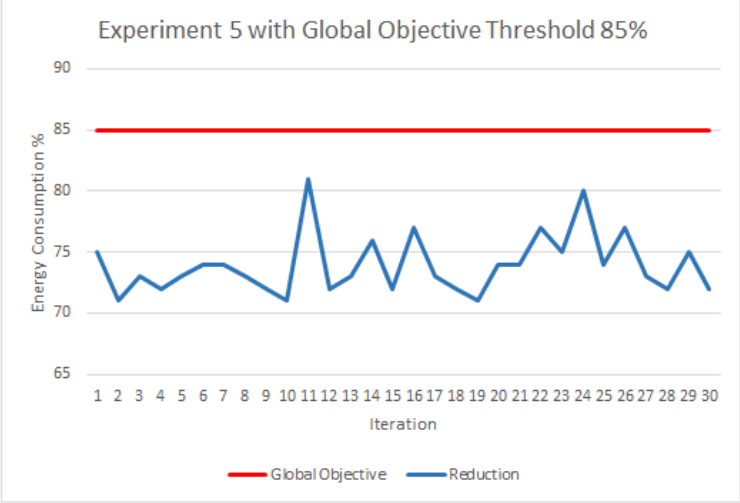


Figure 5.20: Experiment 5 - 85% Global Objective

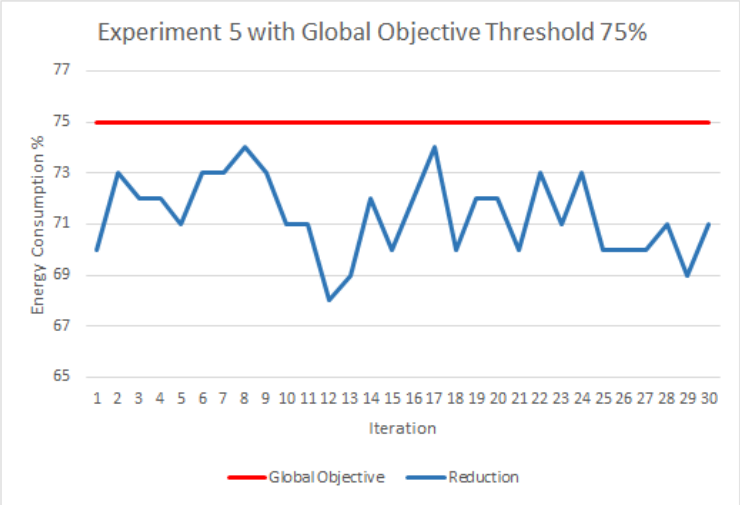


Figure 5.21: Experiment 5 - 75% Global Objective

5.2.7 Experiment 6

The sixth experiment is implementing a larger scale environment, where 1000 houses are used, and there is a random number of rooms and a random number of people in each house. The following are the assumptions for the experiment:

1. Multiple rooms in each house. Each room has a sensor with one or more variables.

2. Each house has its own local objective based on the number of people and the relative energy consumption with respect to the mean, and the desired target consumption.

3. The global objective the city is trying to achieve is to not exceed a threshold limit of 90% of the original consumption which is 2355330 kilowatts. This will reduce the energy consumption by at least 10%.

4. Global objective is changed to have a threshold limit of 85% of the original consumption when the experiment started. This will reduce the energy consumption by at least 15%.

5. Global objective is changed to have a threshold limit of 75% of the original consumption when the experiment started. This will reduce the energy consumption by at least 25%.

6. Global objective is changed to have a threshold limit of 50% of the original consumption.

Results for Experiment 6

The results for Experiment 6 with different global objectives are illustrated in Figures 5.22, 5.23, 5.24, and 5.25. The figures indicate that the energy consumption percentage are below the threshold except when the global objective threshold is 75% and 50%. The consumption levels exceed the limit in some of the iterations when the limit is 75% by a small difference. It can be noticed that the maximum threshold that can be met is 66%, after that the energy consumption cannot be reduced further. For that, when a global objective of a maximum threshold of 50% is introduced, the consumption levels exceeded the limit in all the iterations.

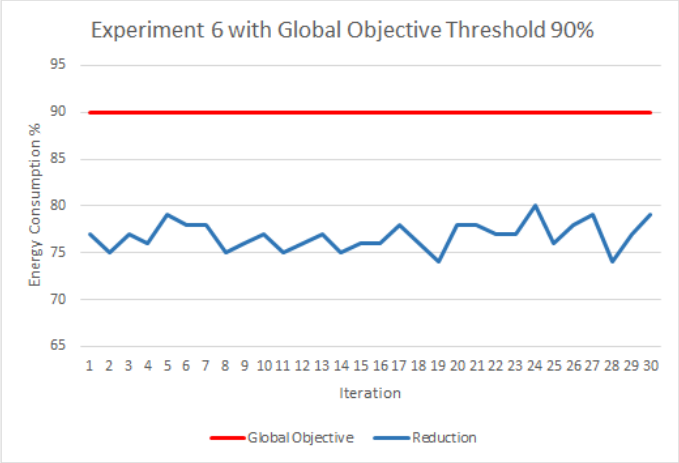


Figure 5.22: Experiment 6 - 90% Global Objective

Moreover, Figure 5.26 shows how the simulation works and provide results. The Figure is related to this final experiment, where the global objective introduced a limit threshold of 75% of the total energy consumption. From the figure, it can be seen that after running the experiment, the model detected the houses that are over the consumption, and is targeted to reduce their consumption by moving them to the class that will achieve that reduction. Also, the energy consumption of smart city is 73.1% now, which means that it did not go over the

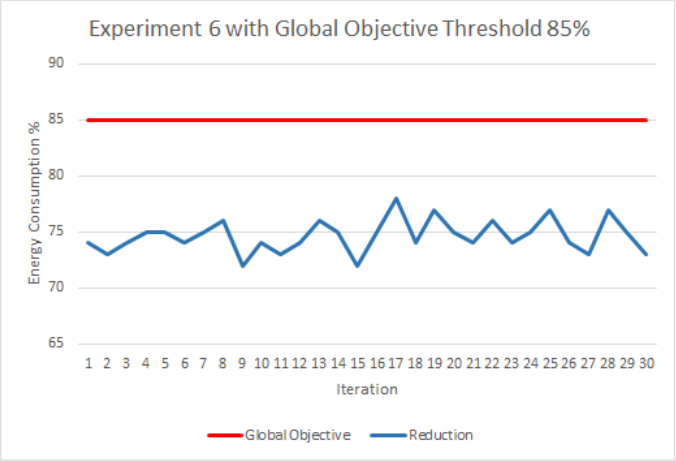


Figure 5.23: Experiment 6 - 85% Global Objective

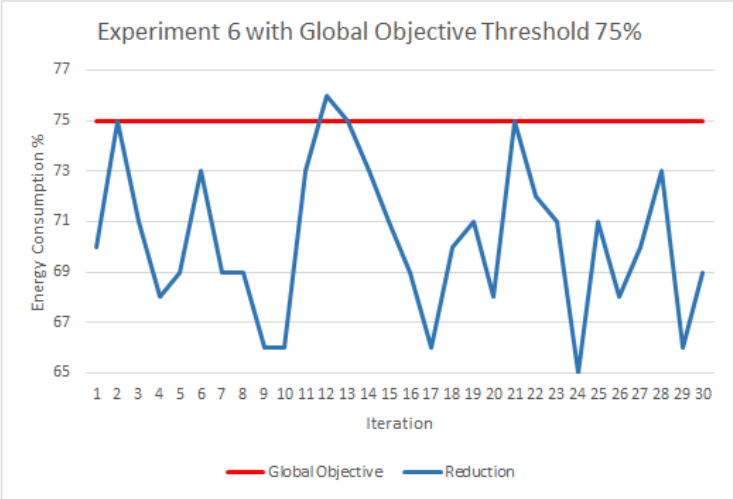


Figure 5.24: Experiment 6 - 75% Global Objective

global objective limit and maintained the restrictions. The new total energy consumption is now 1721009.5 kilowatts, which is 26.9% less than the old consumption of 2355330 kilowatts.

In conclusion, from all the experiments held and the obtained results, the maximum reduced consumption that was achieved is 34%, which means the maximum limit that can imposed is around 66%. Obviously, setting the maximum threshold at certain levels is not always achievable. This experiment indicates that in doing so the system tries to achieve that level but cannot. It does however, result in a reduced level of energy consumption.

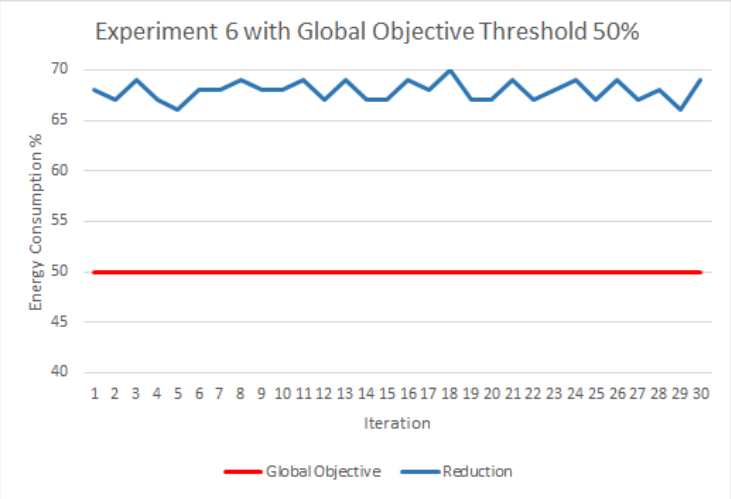


Figure 5.25: Experiment 6 - 50% Global Objective

```
Problems  @ Javadoc  Declaration  Console
<terminated> Smartcity (2) [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe  (Dec 22, 2020, 2:15:24 PM – 2:15:25 PM)
 house 967 has high consumption. It is consuming 5385.0 target is 4038.75 target class 40
 house 969 has high consumption. It is consuming 5610.0 target is 4207.5 target class 42
 house 970 has high consumption. It is consuming 2955.0 target is 2216.25 target class 22
 house 972 has high consumption. It is consuming 4080.0 target is 3060.0 target class 30
 house 974 has high consumption. It is consuming 5460.0 target is 4095.0 target class 40
 house 975 has high consumption. It is consuming 4320.0 target is 3240.0 target class 32
 house 985 has high consumption. It is consuming 2475.0 target is 1856.25 target class 18
 house 986 has high consumption. It is consuming 3105.0 target is 2328.75 target class 23
 house 989 has high consumption. It is consuming 3120.0 target is 2340.0 target class 23
 house 991 has high consumption. It is consuming 3465.0 target is 2598.75 target class 25
 house 996 has high consumption. It is consuming 2685.0 target is 2013.75 target class 20
 house 997 has high consumption. It is consuming 2775.0 target is 2081.25 target class 20

Global Objective Threshold set: 75.0%

The Energy Consumption Percentage of Smart City now:  73.06872073127757%

Old Consumption: 2355330.0 KiloWatt

New Consumption: 1721009.5 KiloWatt
```

Figure 5.26: Output from Simulator for Experiment 6

Furthermore, it can be observed that the percentages of energy consumption are similar since the load is on every house and the decision made is per house. The idea is that we guarantee that it does not exceed the imposed limit. The range set can vary as it is probabilistic and some houses might be empty or some houses have lower energy consumption in general. Since probabilities are used, the energy consumption values of each house is not guaranteed. The important thing is that the global objective is maintained.

Chapter 6

Conclusion and Future Work

In conclusion, the proposed model and algorithms were evaluated by simulating a smart city with different scenarios with the aim to reduce the energy consumption to meet a global objective. The results demonstrated the proposed model and algorithms effectiveness in managing and lowering the energy consumption in a city by around 34%. Six experiments simulated diverse situations where multiple houses with different settings and environments were considered. In each experiment, a global objective that limited the energy consumption in a city was imposed and the energy consumption levels were decreased.

6.1 Summary of Contributions

In this thesis, a model for a smart city physical layer is proposed. The model provided a mathematical model for the framework components of the basic physical infrastructure in a smart city and their relationships with each other. The main components taken into consideration encompass the physical environment and the operational environment. The physical environment model includes the environment $\mathbb{E}$, the sensors $\mathbb{S}$, and the actions Λ, whereas, the operational

environment model includes the intelligent agents $\mathbb{A}$, the reading history $\mathbb{H}$, and the objectives $\mathbb{O}$. A novel semi-supervised Bayesian machine learning algorithm is proposed. The novel algorithm learns the behavior of a sensor to predict the future actions in order to reach a local or global objective. A simulator that simulates the physical layer model and the proposed learning algorithm was programmed and employed to test the model and algorithm.

The main contributions in this thesis are the proposed physical layer framework and the enhanced Bayesian machine learning algorithm, where both can be utilized to leverage smart city services and applications.

6.2 Future Work:

There are some limitations in the proposed work and can be addressed in further extensions. Firstly, the proposed physical model can be utilized to target other objectives in a smart city, such as traffic management and smart streetlight systems. Therefore, a research extension can be conducted to target various objectives besides energy consumption management. Experimentation with larger data sets can be done to explore whether the novel learning algorithm can learn more interesting features or produce better classification and prediction results. Additionally, policies can be identified for different types of managed objects in a smart city, e.g., policies dealing with configuration management and adaptation could be examined, as sensors may fail or stop working at any given time, affecting the whole system or subsystem.

Appendix A

List of Appendices

Figure A.1 illustrates the full class diagram for the simulator discussed in Chapter 5.

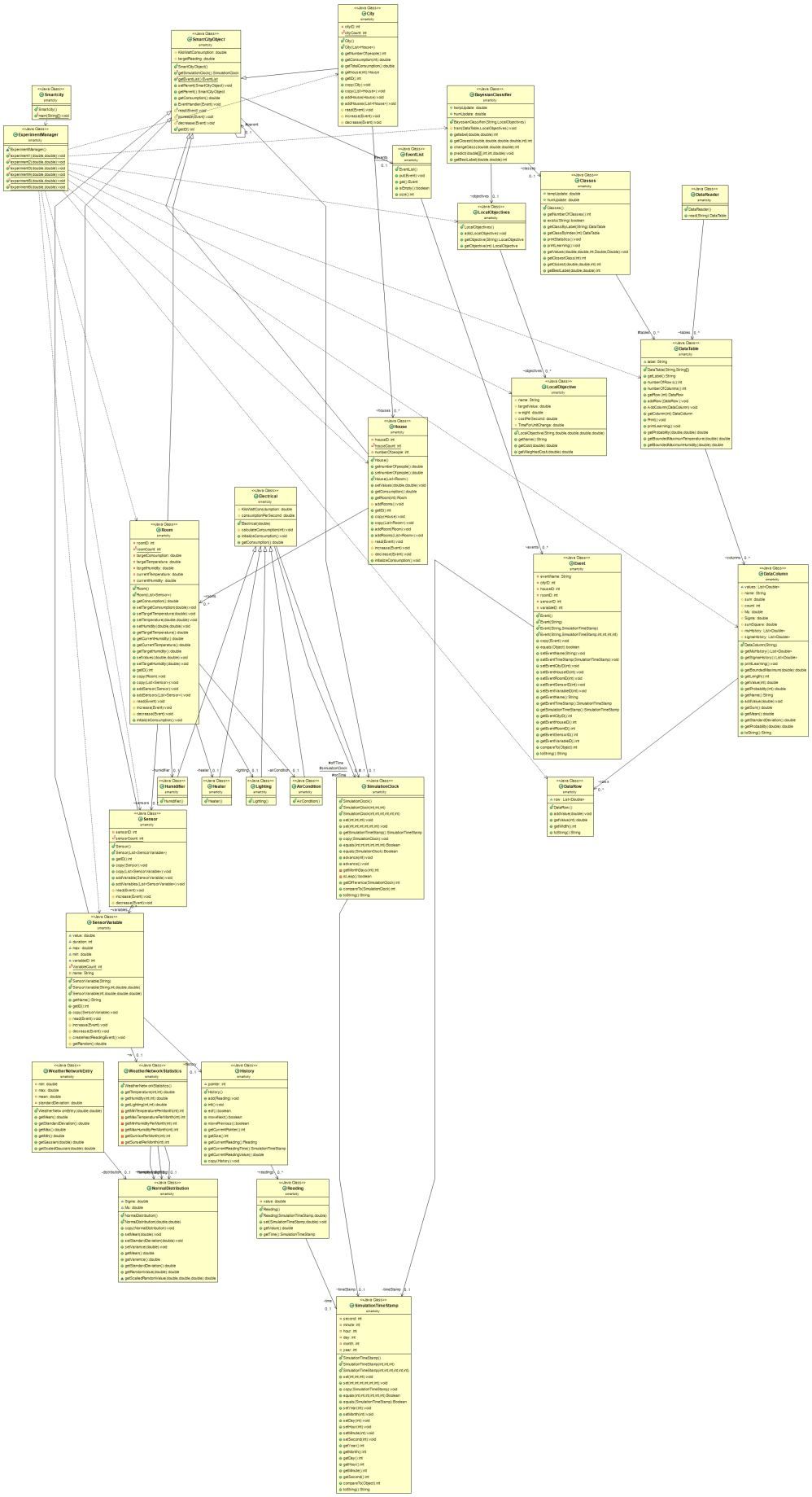


Figure A.1: Class Diagram

Bibliography

[1] United Nations. 68% of the world population projected to live in urban areas by 2050, 2018. https://www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html.

[2] Grand View Research. Smart cities market size, share & trends analysis report, 2020. https://www.grandviewresearch.com/industry-analysis/smart-cities-market.

[3] Irene Celino and Spyros Kotoulas. Smart cities [guest editors' introduction]. *IEEE Internet Computing*, 17(6):8–11, 2013.

[4] Renata Paola Dameri. Searching for smart city definition: a comprehensive proposal. *international Journal of computers & technology*, 11(5):2544–2551, 2013.

[5] Giuseppe Anastasi, Michela Antonelli, Alessio Bechini, Simone Brienza, Eleonora D'Andrea, Domenico De Guglielmo, Pietro Ducange, Beatrice Lazzerini, Francesco Marcelloni, and Armando Segatori. Urban and social sensing for sustainable mobility in smart cities. In *2013 Sustainable Internet and ICT for Sustainability (SustainIT)*, pages 1–4. IEEE, 2013.

[6] José Antonio Galache, Takuro Yonezawa, Levent Gurgen, Daniele Pavia, Marco Grella, and Hiroyuki Maeomichi. Clout: Leveraging cloud computing techniques for improving

management of massive iot data. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, pages 324–327. IEEE, 2014.

[7] Riccardo Petrolo, Valeria Loscri, and Nathalie Mitton. Towards a smart city based on cloud of things, a survey on the smart city vision and paradigms. *Transactions on Emerging Telecommunications Technologies*, 28(1):e2931, 2017.

[8] Kenji Tei and Levent Gürgen. Clout: Cloud of things for empowering the citizen clout in smart cities. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 369–370. IEEE, 2014.

[9] Bin Cheng, Salvatore Longo, Flavio Cirillo, Martin Bauer, and Ernoe Kovacs. Building a big data platform for smart cities: Experience and lessons from santander. In *2015 IEEE International Congress on Big Data*, pages 592–599. IEEE, 2015.

[10] Zaheer Khan, Ashiq Anjum, Kamran Soomro, and Muhammad Atif Tahir. Towards cloud based big data analytics for smart future cities. *Journal of Cloud Computing*, 4(1):1–11, 2015.

[11] Wolfgang Apolinarski, Umer Iqbal, and Josiane Xavier Parreira. The gambas middleware and sdk for smart city applications. In *2014 IEEE International conference on pervasive computing and communication workshops (PERCOM WORKSHOPS)*, pages 117–122. IEEE, 2014.

[12] Chao Wu, David Birch, Dilshan Silva, Chun-Hsiang Lee, Orestis Tsinalis, and Yike Guo. Concinnity: A generic platform for big sensor data applications. *IEEE Cloud Computing*, 1(2):42–50, 2014.

[13] Andreea Claudia Şerban and Miltiadis D Lytras. Artificial intelligence for smart renewable energy sector in europe—smart energy infrastructures for next generation smart cities. *IEEE Access*, 8:77364–77377, 2020.

[14] Laura-Diana Radu. Disruptive technologies in smart cities: A survey on current trends and challenges. *Smart Cities*, 3(3):1022–1038, 2020.

[15] Angelo Cenedese, Andrea Zanella, Lorenzo Vangelista, and Michele Zorzi. Padova smart city: An urban internet of things experimentation. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*, pages 1–6. IEEE, 2014.

[16] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[17] Jeannette Chin, Vic Callaghan, and Ivan Lam. Understanding and personalising smart city services using machine learning, the internet-of-things and big data. In *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, pages 2050–2055. IEEE, 2017.

[18] David V Gibson, George Kozmetsky, and Raymond W Smilor. *The technopolis phenomenon: Smart cities, fast systems, global networks*. Rowman & Littlefield, 1992.

[19] Marcin Dryjanski, Mateusz Buczkowski, Youssouf Ould-Cheikh-Mouhamedou, and Adrian Kliks. Adoption of smart cities with a practical smart building implementation. *IEEE Internet of Things Magazine*, 3(1):58–63, 2020.

[20] Wuu Wen Lin. Novel distributed fiber optic leak system. *Optical Engineering*, 43(2):278–280, 2004.

[21] Antonio Carrillo, Enrique Gonzalez, Armando Rosas, and Alfredo Marquez. New distributed optical sensor for detection and localization of liquid leaks: Part i. experimental studies. *Sensors and Actuators A: Physical*, 99(3):229–235, 2002.

[22] Vehbi C Gungor, Bin Lu, and Gerhard P Hancke. Opportunities and challenges of wireless sensor networks in smart grid. *IEEE transactions on industrial electronics*, 57(10):3557–3564, 2010.

[23] Tianyu Zhang, Qian Zhao, Kilho Shin, and Yukikazu Nakamoto. Bayesian-optimization-based peak searching algorithm for clustering in wireless sensor networks. *Journal of Sensor and Actuator Networks*, 7(1):2, 2018.

[24] Wen-Tsai Sung. Multi-sensors data fusion system for wireless sensors networks of factory monitoring via bpn technology. *Expert Systems with Applications*, 37(3):2124–2131, 2010.

[25] Gregory Hackmann, Weijun Guo, Guirong Yan, Zhuoxiong Sun, Chenyang Lu, and Shirley Dyke. Cyber-physical codesign of distributed structural health monitoring with wireless sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 25(1):63–72, 2013.

[26] Luís ML Oliveira and Joel JPC Rodrigues. Wireless sensor networks: A survey on environmental monitoring. *JCM*, 6(2):143–151, 2011.

[27] Che-I Wu, Hsu-Yang Kung, Chi-Hua Chen, and Li-Chia Kuo. An intelligent slope disaster prediction and monitoring system based on wsn and anp. *Expert Systems with Applications*, 41(10):4554–4562, 2014.

[28] Jin Wang, Zhongqi Zhang, Bin Li, Sungyoung Lee, and R Simon Sherratt. An enhanced fall detection system for elderly person monitoring using consumer home networks. *IEEE transactions on consumer electronics*, 60(1):23–29, 2014.

[29] João Cunha, Nelson Batista, Carlos Cardeira, and Rui Melicio. Wireless networks for traffic light control on urban and aerotropolis roads. *Journal of Sensor and Actuator Networks*, 9(2):26, 2020.

[30] Mohd Fauzi Othman and Khairunnisa Shazali. Wireless sensor network applications: A study in environment monitoring system. *Procedia Engineering*, 41:1204–1210, 2012.

[31] Ivanovitch Silva, Luiz Affonso Guedes, Paulo Portugal, and Francisco Vasques. Reliability and availability evaluation of wireless sensor networks for industrial applications. *Sensors*, 12(1):806–838, 2012.

[32] Gang Zhao et al. Wireless sensor networks for industrial process monitoring and control: A survey. *Netw. Protoc. Algorithms*, 3(1):46–63, 2011.

[33] Byeongkwan Kang, Seunghwan Kim, Myeong-in Choi, Keonhee Cho, Seongman Jang, and Sehyun Park. Analysis of types and importance of sensors in smart home services. In *2016 IEEE 18th International Conference on High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 1388–1389. IEEE, 2016.

[34] Eduardo Felipe Zambom Santana, Ana Paula Chaves, Marco Aurelio Gerosa, Fabio Kon, and Dejan S Milojicic. Software platforms for smart cities: Concepts, requirements, challenges, and a unified reference architecture. *ACM Computing Surveys (Csur)*, 50(6):1–37, 2017.

[35] Giuseppe Piro, Ilaria Cianci, Luigi Alfredo Grieco, Gennaro Boggia, and Pietro Camarda. Information centric services in smart cities. *Journal of Systems and Software*, 88:169–188, 2014.

[36] Muhammad Iqbal, Muhammad Naeem, Alagan Anpalagan, Ashfaq Ahmed, and Muhammad Azam. Wireless sensor network optimization: Multi-objective paradigm. *Sensors*, 15(7):17572–17620, 2015.

[37] Daniel Minoli, Kazem Sohraby, and Benedict Occhiogrosso. Iot considerations, requirements, and architectures for smart buildings—energy optimization and next-generation building management systems. *IEEE Internet of Things Journal*, 4(1):269–283, 2017.

[38] Hisham Albataineh, Mais Nijim, and Divya Bollampall. The design of a novel smart home control system using smart grid based on edge and cloud computing. In *2020 IEEE 8th International Conference on Smart Energy Grid Engineering (SEGE)*, pages 88–91. IEEE, 2020.

[39] Abhishek Bhati, Michael Hansen, and Ching Man Chan. Energy conservation through smart homes in a smart city: A lesson for singapore households. *Energy Policy*, 104:230–239, 2017.

[40] Segun O Olatinwo and Trudi-H Joubert. Optimizing the energy and throughput of a water-quality monitoring system. *Sensors*, 18(4):1198, 2018.

[41] Guwon Yoon, Sanguk Park, Sangmin Park, Tacklim Lee, Seunghwan Kim, Hyeonwoo Jang, Sanghyeon Lee, and Sehyun Park. Prediction of machine learning base for efficient use of energy infrastructure in smart city. In *2019 International Conference on Computing, Electronics & Communications Engineering (iCCECE)*, pages 32–35. IEEE, 2019.

[42] Waleed Ejaz, Muhammad Naeem, Adnan Shahid, Alagan Anpalagan, and Minho Jo. Efficient energy management for the internet of things in smart cities. *IEEE Communications Magazine*, 55(1):84–91, 2017.

[43] Tao Hong and Shu Fan. Probabilistic electric load forecasting: A tutorial review. *International Journal of Forecasting*, 32(3):914–938, 2016.

[44] Borje Ohlman, Anders Eriksson, and Rene Rembarz. What networking of information can do for cloud computing. In *2009 18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, pages 78–83. IEEE, 2009.

[45] Carl Shapiro, Shapiro Carl, Hal R Varian, et al. *Information rules: a strategic guide to the network economy*. Harvard Business Press, 1998.

[46] Joanna Gordon, Chiemi Hayashi, Dan Elron, L Huang, and R Neill. Exploring the future of cloud computing: riding the next wave of technology-driven transformation. In *World Economic Forum*, 2010.

[47] Andy Davis, Jay Parikh, and William E Weihl. Edgecomputing: extending enterprise applications to the edge of the internet. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 180–187, 2004.

[48] Samee U Khan. The curious case of distributed systems and continuous computing. *IT Professional*, 18(2):4–7, 2016.

[49] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[50] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16, 2012.

[51] Prabal Verma and Sandeep K Sood. Fog assisted-iot enabled patient health monitoring in smart homes. *IEEE Internet of Things Journal*, 5(3):1789–1796, 2018.

[52] Tom H Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. Fog computing: Focusing on mobile users at the edge. *arXiv preprint arXiv:1502.01815*, 2015.

[53] Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, and Tie Qiu. Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of network and computer applications*, 98:27–42, 2017.

[54] Quan Zhang, Xiaohong Zhang, and Weisong Shi. Firework: big data processing in collaborative edge environment. In *2016 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 81–82. IEEE, 2016.

[55] Yahya Mohamed. What is the role of iot in smart cities?, Sep 2019.

[56] Andrea Zanella, Nicola Bui, Angelo Castellani, Lorenzo Vangelista, and Michele Zorzi. Internet of things for smart cities. *IEEE Internet of Things journal*, 1(1):22–32, 2014.

[57] Saber Talari, Miadreza Shafie-Khah, Pierluigi Siano, Vincenzo Loia, Aurelio Tommasetti, and João PS Catalão. A review of smart cities based on the internet of things concept. *Energies*, 10(4):421, 2017.

[58] Alessio Botta, Walter De Donato, Valerio Persico, and Antonio Pescapé. Integration of cloud computing and internet of things: a survey. *Future generation computer systems*, 56:684–700, 2016.

[59] Saima Zafar, Ghosia Miraj, Rajaa Baloch, Danish Murtaza, and Khadija Arshad. An iot based real-time environmental monitoring system using arduino and cloud service. *Engineering, Technology & Applied Science Research*, 8(4):3238–3242, 2018.

[60] Anand Nayyar and Vikram Puri. Smart farming: Iot based smart sensors agriculture stick for live temperature and moisture monitoring using arduino, cloud computing & solar technology. In *Proc. of The International Conference on Communication and Computing Systems (ICCCS-2016)*, pages 9781315364094–121, 2016.

[61] Guigang Zhang, Chao Li, Yong Zhang, Chunxiao Xing, and Jijiang Yang. Semanmedical: A kind of semantic medical monitoring system model based on the iot sensors. In *2012 IEEE 14th international conference on e-health networking, applications and services (Healthcom)*, pages 238–243. IEEE, 2012.

[62] Aamir Nizam Ansari, Mohamed Sedky, Neelam Sharma, and Anurag Tyagi. An internet of things approach for motion detection using raspberry pi. In *Proceedings of 2015 International Conference on Intelligent Computing and Internet of Things*, pages 131–134. IEEE, 2015.

[63] R Deekshath, P Dharanya, Ms KR Dimpil Kabadia, Mr G Deepak Dinakaran, and S Shanthini. Iot based environmental monitoring system using arduino uno and thingspeak. *International Journal of Science Technology & Engineering*, 4(9), 2018.

[64] Ravi Kishore Kodali, Vishal Jain, Suvadeep Bose, and Lakshmi Boppana. Iot based smart security and home automation system. In *2016 international conference on computing, communication and automation (ICCCA)*, pages 1286–1289. IEEE, 2016.

[65] Y Raghavender Rao. Automatic smart parking system using internet of things (iot). *Int J Eng Technol Sci Res*, 4(5), 2017.

[66] Abhimanyu Singh, Pankhuri Aggarwal, and Rahul Arora. Iot based waste collection system using infrared sensors. In *2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pages 505–509. IEEE, 2016.

[67] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[68] Irshad Ullah, MNR Baharom, H Ahmad, Fazli Wahid, HM Luqman, Zainab Zainal, and B Das. Smart lightning detection system for smart-city infrastructure using artificial neural network. *Wireless Personal Communications*, 106(4):1743–1766, 2019.

[69] Dohyung Kim, Hyochang Yang, Minki Chung, Sungzoon Cho, Huijung Kim, Minhee Kim, Kyungwon Kim, and Eunseok Kim. Squeezed convolutional variational autoencoder for unsupervised anomaly detection in edge device industrial internet of things. In *2018 international conference on information and computer technologies (icict)*, pages 67–71. IEEE, 2018.

[70] Dong Yul Oh and Il Dong Yun. Residual error based anomaly detection using autoencoder in smd machine sound. *Sensors*, 18(5):1308, 2018.

[71] Propagation Fusion. Structuring in belief networks. Technical report, Tech. Report 3, UCLA Computer Science Department, 1986, Technical Report, 1986.

[72] Xin Wang and Peng Guo. A novel binary adaptive differential evolution algorithm for bayesian network learning. In *2012 8th International Conference on Natural Computation*, pages 608–612. IEEE, 2012.

[73] Artur Arsénio, Hugo Serra, Rui Francisco, Fernando Nabais, João Andrade, and Eduardo Serrano. Internet of intelligent things: Bringing artificial intelligence into things and communication networks. In *Inter-cooperative collective intelligence: Techniques and applications*, pages 1–37. Springer, 2014.

[74] The Weather Network Pelmorex Weather Networks. Toronto, on weather, 2020. `https://www.theweathernetwork.com/ca/14-day-weather-trend/ontario/toronto`.

[75] Stamatis Karnouskos and Thiago Nass De Holanda. Simulation of a smart grid city with software agents. In *2009 Third UKSim European Symposium on Computer Modeling and Simulation*, pages 424–429. IEEE, 2009.

Curriculum Vitae

Name:	Razan Essam AlFar
Post-Secondary Education and Degrees:	American University of the Middle East (AUM) Egaila, Kuwait 2014 - 2018 B.Sc. Western University London, ON 2019 - 2021 M.Sc.
Honours and Awards:	AUM Honor's Scholarship 2014-2018 Western Graduate Research Scholarship (WGRS) 2019-2020
Related Work Experience:	Teaching and Research Assistant Western University 2019 - 2020

Publications:

R. AlFar, Y. Kotb, and M. Bauer, "Controlling and Optimizing Sensor Network in Smart Cities Using Intelligent Agents," to appear in 2021 Future of Information and Communication Conference (FICC), Vancouver, BC, Canada, Apr. 2021.